



저작자표시-비영리-변경금지 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.

다음과 같은 조건을 따라야 합니다:



저작자표시. 귀하는 원저작자를 표시하여야 합니다.



비영리. 귀하는 이 저작물을 영리 목적으로 이용할 수 없습니다.



변경금지. 귀하는 이 저작물을 개작, 변형 또는 가공할 수 없습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 [이용허락규약\(Legal Code\)](#)을 이해하기 쉽게 요약한 것입니다.

[Disclaimer](#)

Master's Thesis

Development of Real-time Image Processing for Embedded System

Soyoon Kim

Department of Mechanical Engineering

Graduate School of UNIST

2019

Development of Real-time Image Processing for Embedded System

Soyoon Kim

Department of Mechanical Engineering

Graduate School of UNIST

Development of Real-time Image Processing for Embedded System

A thesis
submitted to the Graduate School of UNIST
in partial fulfillment of the
requirements for the degree of
Master of Science

Soyoon Kim

12. 13. 2018

Approved by

A handwritten signature in black ink, appearing to read 'H. Son', is written over a horizontal line.

Advisor

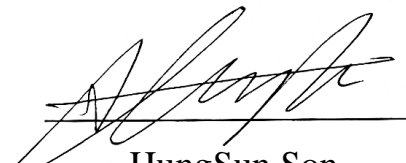
HungSun Son

Development of Real-time Image Processing for Embedded System

Soyoon Kim

This certifies that the thesis of Soyoon Kim is approved


12. 13. 2018



HungSun Son



JoonBum Bae



Hyondong Oh

ABSTRACT

Unmanned aerial vehicles (UAVs) are widely used in various areas such as exploration, transportation and rescue activity due to light weight, low cost, high mobility and intelligence. This intelligent system consists of highly integrated and embedded systems along with a microprocessor to perform specific task by computing algorithm or processing data. In particular, image processing is one of main core technologies to handle important tasks such as target tracking, positioning, visual servoing using visual system. However, it often requires heavy amount of computation burden and an additional micro PC controller with a flight computer should be additionally used to process image data. However, performance of the controller is not so good enough due to limited power, size, and weight. Therefore, efficient image processing techniques are needed considering computing load and hardware resources for real time operation on embedded systems.

The objective of the thesis research is to develop an efficient image processing framework on embedded systems utilizing neural network and various optimized computation techniques to satisfy both efficient computing speed versus resource usage and accuracy. Image processing techniques has been proposed and tested for management computing resources and operating high performance missions in embedded systems. Graphic processing units (GPUs) available in the market can be used for parallel computing to accelerate computing speed. Multiple cores within central processing units (CPUs) are used like multi-threading during data uploading and downloading between the CPU and the GPU. In order to minimize computing load, several methods have been proposed. The first method is visualization of convolutional neural network (CNN) that can perform both localization and detection simultaneously. The second is region proposal for input area of CNN through simple image processing, which helps algorithm to avoid full frame processing. Finally, surplus computing resources can be saved by control the transient performance such as the FPS limitation.

These optimization methods have been experimentally applied to a ground vehicle and quadrotor UAVs and verified that the developed methods offer an optimization to process in embedded environment by saving CPU and memory resources. In addition, they can support to perform various tasks such as object detection and path planning, obstacle avoidance. Through optimization and algorithms, they reveal a number of improvements for the embedded system compared to the existing. Considering the characteristics of the system to transplant the various useful algorithms to the embedded system, the method developed in the research can be further applied to various practical applications.

CONTENTS

ABSTRACT.....	1
CONTENTS.....	2
LIST OF FIGURES	3
LIST OF TABLES.....	5
NOMENCLATURE	6
I. INTRODUCTION.....	7
1.1 Motivation	7
1.2 Related works	8
1.3 Research objective.....	12
II. OPTIMIZATION TECHNIQUES FOR IMAGE PROCESSING	13
2.1 General Purpose of Graphical Processing Units.....	13
2.2 Limitation of Overperformance	18
2.3 Convolutional Neural Network Visualization	19
2.4 Region proposal	31
III. IMPLEMENTATION OF UAVS.....	35
3.1 Setpoint Generation	36
3.2 Collision Potential Area.....	39
3.3 Odometry Assistant using Stationary Landmark	41
IV. CONCLUSION.....	44
REFERENCES	45
ACKNOWLEDGEMENT	47

LIST OF FIGURES

Fig 1.1	UAVs for various purpose.....	7
Fig 1.2	Designed Gabor filters.....	9
Fig 1.3	Different detection qualities depend on threshold	9
Fig 1.4	Structure of CNN.....	10
Fig 1.5	Structure of fully connected layer.....	10
Fig 1.6	Model of neuron inside of network	10
Fig 1.7	Detection of crack with sliding window CNN	11
Fig 2.1	Summary of second section.....	13
Fig 2.2	Comparison CPU of processing on GPGPU(red) CPU only(blue)	14
Fig 2.3	Comparison memory usage of processing on GPGPU(red) CPU only(blue).....	14
Fig 2.4	Saliency detection example	15
Fig 2.5	Program execution using CPU only	16
Fig 2.6	Program execution using CPU and GPU.....	16
Fig 2.7	Total usage of CPU for each test condition	17
Fig 2.8	Memory usage for each test condition.....	17
Fig 2.9	Total usage of CPU at limited performance	18
Fig 2.10	Memory usage at limited performance	18
Fig 2.11	Regression area built by multiple layers.....	19
Fig 2.12	Network structure of trained CNN	21
Fig 2.13	Error drop during network training.....	21
Fig 2.14	Accuracy check for training dataset	22
Fig 2.15	Accuracy check for test dataset	22
Fig 2.16	Visualization of Convolved image with trained kernels.....	24
Fig 2.17	Convergence time with number of neurons.....	25
Fig 2.18	Convolution kernel visualization.....	25
Fig 2.19	Class activation mapping.....	26
Fig 2.20	Level of activation according to the depth of the network	27
Fig 2.21	Crack detection process	29
Fig 2.22	Discrimination crack.....	30
Fig 2.23	ROI proposal as input of CNN	31
Fig 2.24	Test result(1) filtering.....	33
Fig 2.25	Test result(2) filtering and detection.....	33

Fig 2.26	Test result(3) rotation invariant and fault result	34
Fig 2.27	Result of proposed algorithm with various kernels	34
Fig 2.28	CPU total usage [%]	34
Fig. 3.1	Jetson TX1	35
Fig. 3.2	ZED Stereo camera.....	35
Fig. 3.3	Pixhawk	35
Fig. 3.4	Test platform setup	36
Fig. 3.5	Optical motion capture system	36
Fig. 3.6	Depth data correction.....	36
Fig. 3.7	Visual change according to camera position	37
Fig. 3.8	Setpoint calculation using depth data	37
Fig. 3.9	Yaw state data and setpoint	38
Fig. 3.10	Error of generated setpoint	39
Fig. 3.11	Drone size in pixel by depth	40
Fig. 3.12	Processed image with collision potential area and object information.....	40
Fig. 3.13	Processed image with collision potential area and avoiding box	40
Fig. 3.14	UAV's body NED frame	41
Fig. 3.15	Image adjustment for rotation by x-axis	42
Fig. 3.16	Image adjustment for rotation by y-axis	42
Fig. 3.17	Odometry assistant with object as landmark	43

LIST OF TABLES

Table 2.1 Performance and CPU usage	15
Table 2.2 Network design principle.....	19
Table 2.3 Dataset specification.....	20
Table 2.4 Network parameters of trained CNN.....	20
Table 2.5 Procedure of crack detection	28
Table 2.6 Network condition	32
Table 2.7 Training condition	32
Table 2.8 Processing time on real-time CNN.....	33
Table 3.1 Jetson TX1 Specification.....	35
Table 3.2 ZED Stereo Camera Specification.....	35
Table 3.3 Pixhawk Specification	35

NOMENCLATURE

Upper characters

A	Area
C	Contour
E	Error
R	Rotation matrix
I	Image

Lower characters

b	Body frame
p	Pixel value
r	Radius
w	World frame

Greek characters

ϕ	Attitude of roll
θ	Attitude of pitch
ψ	Attitude of yaw
v	Linear velocity
ω	Angular velocity

Abbreviation

CPU	Central Processing Unit
GPU	Graphical Processing Unit
GPGPU	General Purpose Graphical Process Unit
CNN	Convolutional Neural Network
ROI	Region Of Interest
FPS	Frame Per Second
IMU	Inertial Measurement Unit

I. INTRODUCTION

1.1 Motivation

An embedded system on unmanned systems such as aerial vehicle, ground, and underwater, requires processing a large amount of data for a path planning, image processing, various mission in real time. Recently, advanced technologies in computational processing performance, network communication and energy technology have expanded operating area of the unmanned vehicle as well as increasing utilization in various areas such as exploration, transportation and military in Fig 1.1 [1,2]. In addition, various studies have been conducted in the fields of navigation, positioning, communication, power system to meet the system requirements according to the operating plan.

Unmanned aerial vehicles (UAVs) are dramatically getting an attention in a number of applications but the size and weight of the vehicle is still limited by the scale of the operation [3], and the payload is also limited as the thrust is limited. This leads to a limitation of the computing resources being loaded, so it is essential to simplify the algorithm and increase efficiency in UAV operation.



(a) Radiation monitoring



(b) Firefighting



(c) Transportation

Fig 1.1 UAVs for various purpose

In particular, this feature is remarkable in the image processing. Image processing is the core technology to handle important tasks such as target tracking, positioning, data streaming for operating unmanned vehicles. However, the amount of computation depends on the resolution of frame and complexity of the algorithm. It can be a threat to real-time operation on the embedded system.

Convolution neural network (CNN) also can effectively solve complex nonlinear problems specially on image recognition though combination between feature extraction by convolution and classification for prediction by fully connected layer [4] but also, it is forced to bear computing load due to the increase in the calculations by nodes. So, embedded system should minimize computational load and use limited computational resources efficiently.

1.2 Related works

A number of researches have been proposed for performing specific task on unmanned vehicle. Among them, this work will deal with the researches related to target recognition. Target recognition contains object detection and classification, it is much useful technique for autonomous landing [5], autonomous driving [6], inspection [7,8] and so many applications. For example, in the case of detecting cracks through machine vision, there are method of image processing techniques using various kinds of filters such as morphological filter and Gabor filter. Data processing is done by applying Gabor filter invariant to rotation, allowing the detection of cracks in any direction [7]. However, the quality of the result varies greatly depending on the shape of the filter, so the design of the filter becomes very important.

Various segmentation methods can be applied for crack detection, pixel based segmentation represented by region growing, have same limitation which difficult to choose number of seed pixels and effective position of seed pixel and design growing mask.

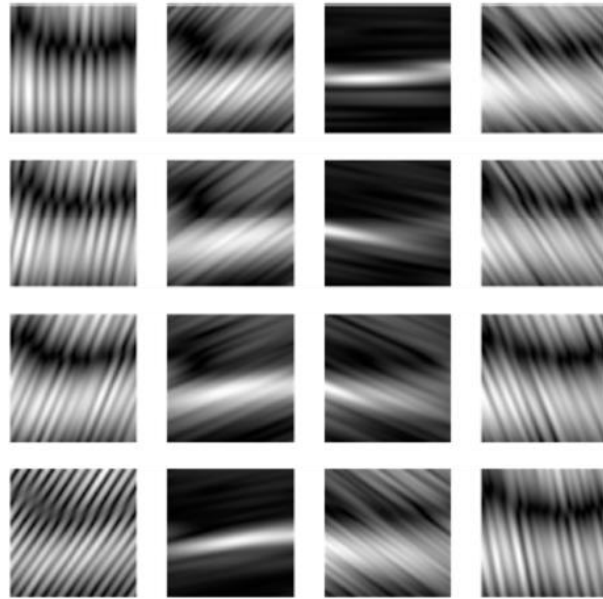


Fig 1.2 Designed Gabor filters [7]

Similarly, threshold based approaches may not be a general solution, depending on the designer's perspective and the propensity of the data, because the quality of the results depends entirely on the threshold [9]. Due to these limitations, researches have proposed that do not require thresholds or that leave the design of filters to be learned artificial intelligence based on data.

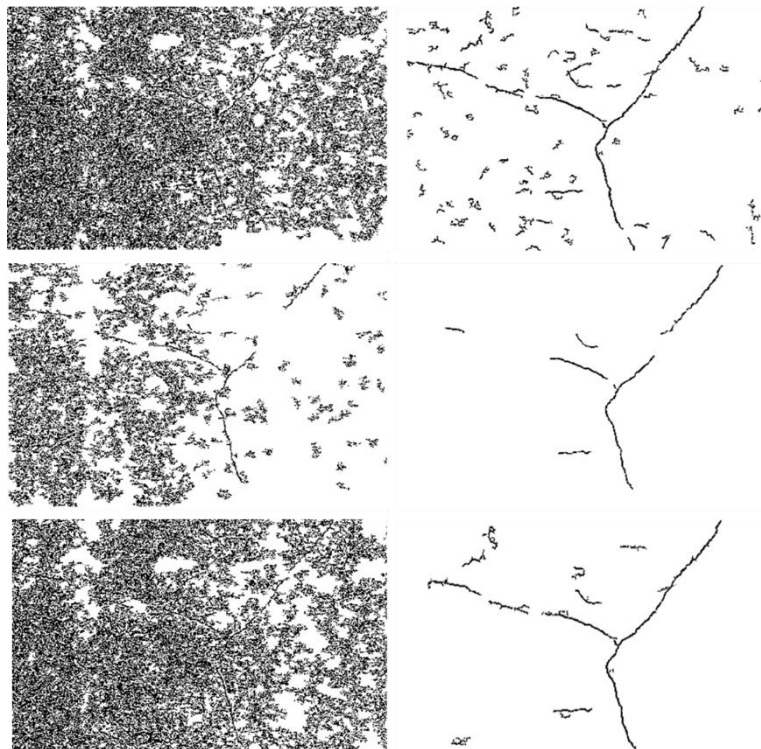


Fig 1.3 Different detection qualities depend on threshold [9]

Artificial intelligence rapidly becomes popular as a powerful technology to solve the various nonlinear problems that exist in real life. Especially, voice recognition, image recognition, and others are used in areas where there is a lot of noise and regularity cannot be derived. In the case of the artificial neural network, one of the deep learning technologies, the linear equation-based nodes are complex in structures like human neurons, creating nonlinear systems. It can draw multi regression area.

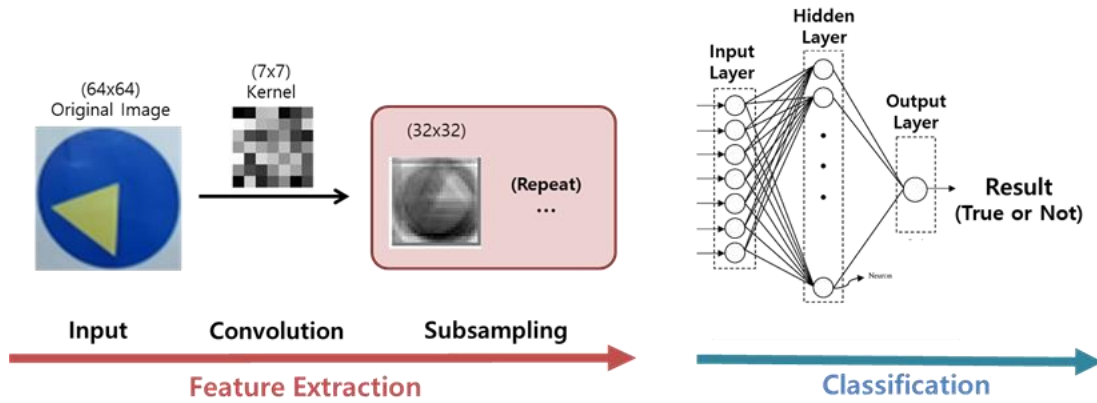


Fig 1.4 Structure of CNN

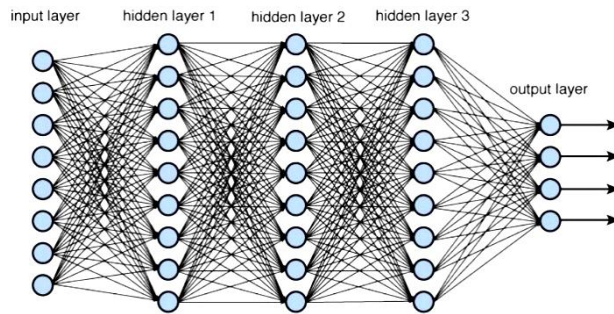


Fig 1.5 Structure of fully connected layer

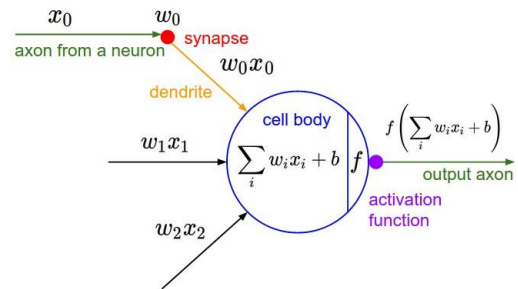


Fig 1.6 Model of neuron inside of network

Convolution is usually divided two steps, convolution layer and fully connected layer. The first step is for extraction characteristics by emphasizing specific parts of the image. And the next, with convolved image, network predicts output as result of calculation by nodes on fully connected layer. Both the applied kernels of convolution layer and the weights of fully connected layer are used as parameters learned through the backpropagation process by the error value from training dataset.

The results obtained from CNN features and nonlinearities show good performance in object recognition. In the case of the crack detection mentioned above, results were also shown [8].

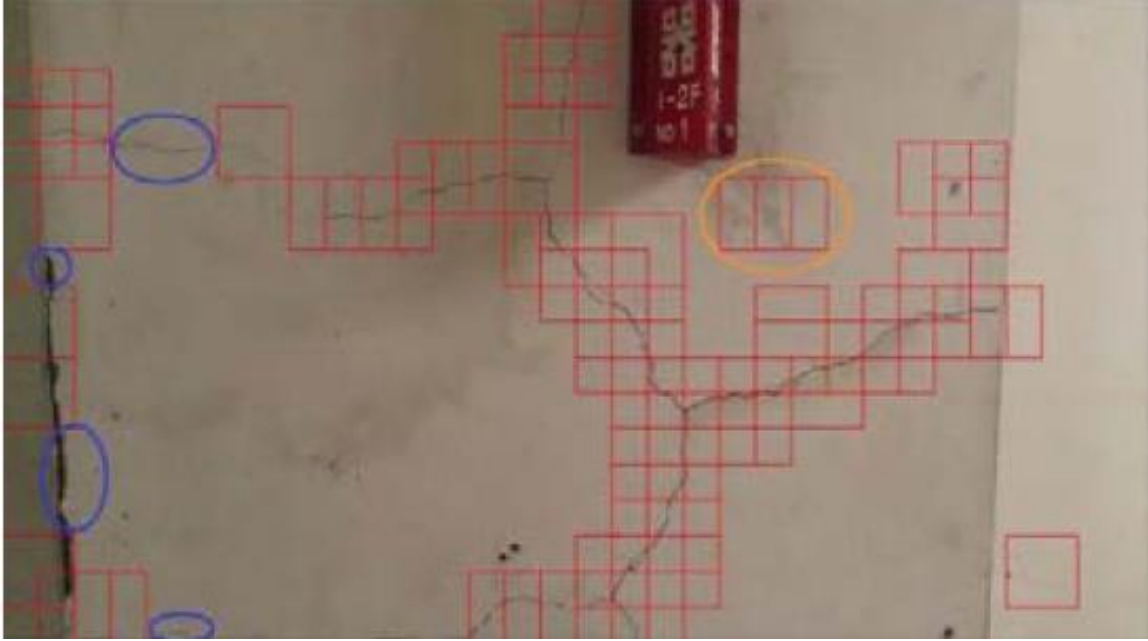


Fig 1.7 Detection of crack with sliding window CNN [8]

However, most of CNN algorithms ask high computational cost because of repetitive computation in convolution with kernels and linear computation with nodes. As shown in Fig 1.7, the researchers used that the sliding window method shown result, which requires a lot of computation by searching all areas of the frame. In the paper, we can confirm that the hardware specification is not mentioned but it is processed at 40s per frame. Frame Per Second (FPS) is under 0.025.

To overcome the limitations of processing speed due to repeated computations, General Purpose GPU (GPGPU) became popular to apply parallel computation. GPGPU is computing acceleration technique using Graphical Processing Unit (GPU) which is a device originally developed for graphical work and is specialized in parallel computing, consisting of hundreds to thousands of processing units. With the development of artificial intelligence, the computational quantity has increased rapidly and with the use of a parallel computation capable GPU. The GPU's utilization area is expanded from simple graphical tasks to general purpose computing processors. Due to this usefulness, NVIDIA developed a variety of open source API, CUDA, for developers who want to use GPGPU and released the Jetson series to build ecosystems in embedded systems.

Some algorithms suggested an optimized structure for real-time application of embedded systems [10,11], but still most of the algorithms cannot reach a significant speed of processing, at less than FPS 3. Recent researches have been started to consider real-time operation on embedded environment but they concentrate on processing speed only [12,13]. There are not any mention about computing load only performance. So, there are still lack relevant studies on real world application.

1.3 Research objective

The objective of this research is to develop effective image processing framework on embedded systems using neural network and various optimized computation techniques.

There are three ways to minimize computational load. First is GPGPU, it helps by computing acceleration. Secondly, simplified structure from basis and high robustness are required for high performance and efficiency. For example, CNN algorithms is designed optimized and simplified through ROI pre-processing. Finally, the surplus resources are saved by limiting the processing speed as much as much as possible except desired performance.

These techniques can be used to verify the effects in the real-time processing and tested in the operating environment of a quadrotor type drone.

II. OPTIMIZATION TECHNIQUES FOR IMAGE PROCESSING

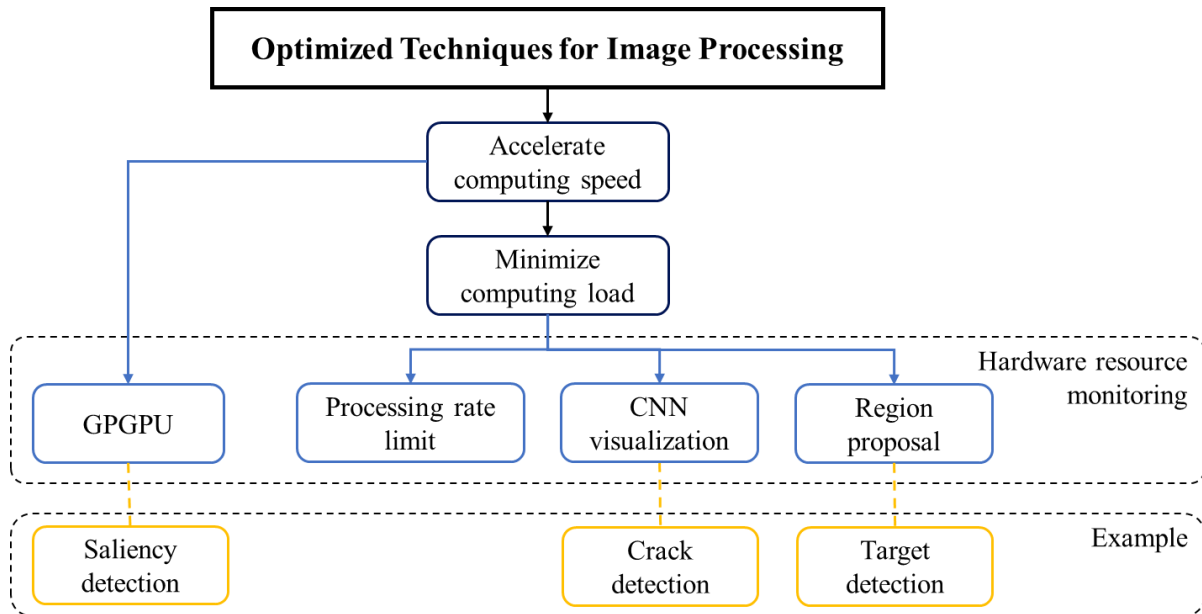


Fig 2.1 Summary of second section

2.1 General Purpose of Graphical Processing Units

In this part, the practical method will be proposed such as utilizing GPU, saving surplus computing resources, and usage on each core of Central Processing Unit (CPU), total usage of CPU and memory usage. To overcome the limitations of Open API, customizable program was built using C++ programs to ensure free and fast performance and image processing was implemented using OpenCV, an open source library on image processing.

Although the tools provided by NVIDIA may be used directly for GPU programming, the programs developed in this study used the CUDA modules provided by the OpenCV library. Unlike CPU, there is no control unit or data cache, so memory occupancy occurs when GPU forwards processed data back to the CPU. So, it is needed to memory management and scheduling of each unit's work because if memory shortage occurred, every process stops and delayed until it recovers spare memory space. In Fig 2.2 and Fig 2.3, red line shows running by CPU only and blue line by GPU maximized running. With graph which presents monitoring result, CPU usage was decreased during GPU running, but also memory usage was increased and kept even after end of the program.

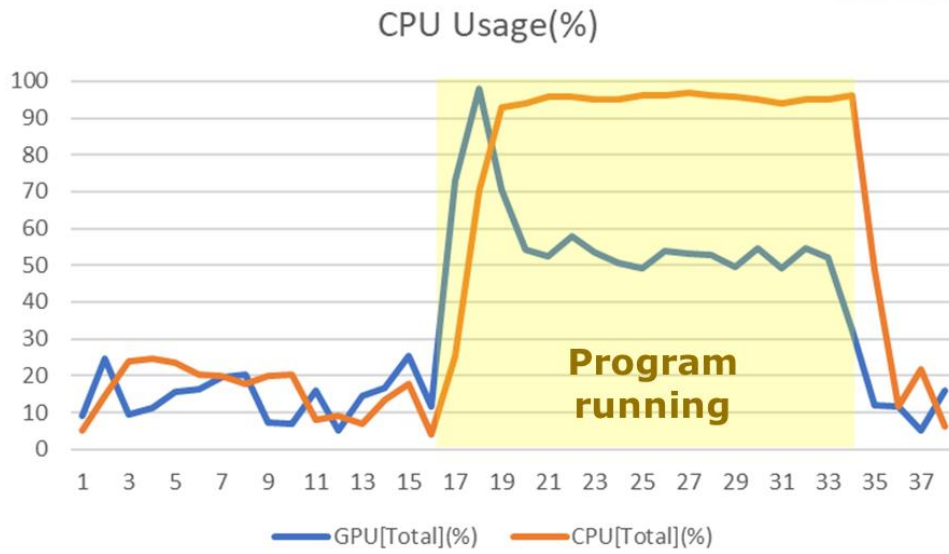


Fig 2.2 Comparison CPU of processing on GPGPU(red) CPU only(blue)

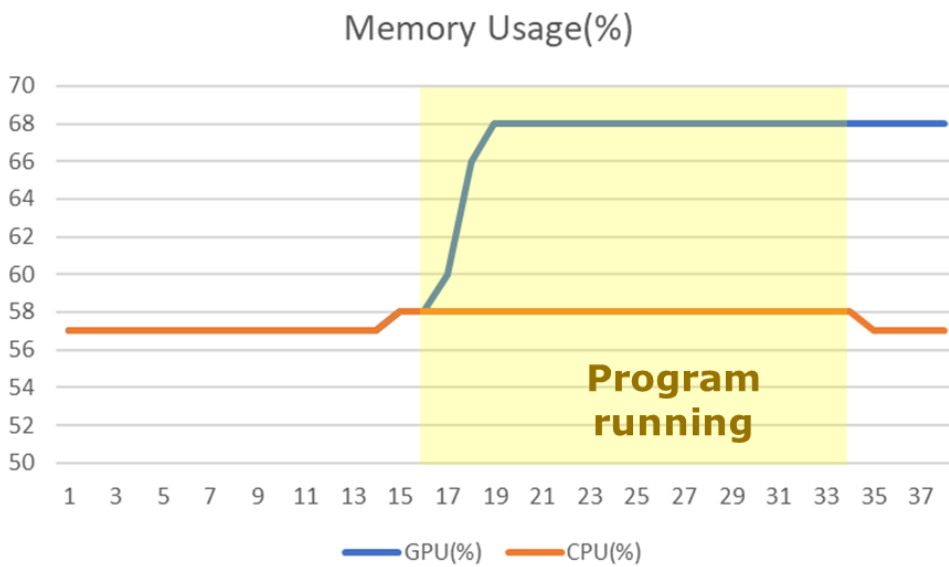
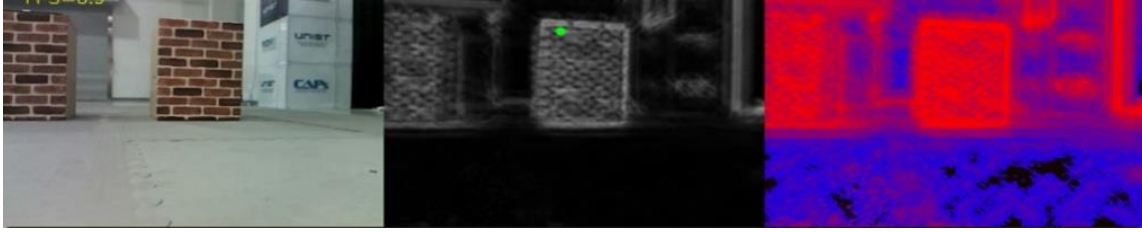


Fig 2.3 Comparison memory usage of processing on GPGPU(red) CPU only(blue)

Saliency detection[14] was tested to apply GPGPU on embedded environment. Result was drawn in Fig 2.4 and summarized in the Table 2.1. The CPU model of embedded board is ARM Cortex-A57 (quad-core) @ 1.73GHz.



(a) Result image without CUDA processing



(b) Result image with CUDA processing

Fig 2.4 Saliency detection example

Table 2.1 Performance and CPU usage

	CPU only	GPU maximize	GPU optimized
FPS	2.1	4.4~4.6	4.4~4.6
CPU avg [%]	25.97	29.38	27.12

Another phenomenon that occurs when using GPUs is the behavior of each CPU core differentiates appeared in Fig 2.5 and Fig 2.6. Only single thread is running, only one core is responsible for computing at its maximum without GPGPU. The other case is distributed computing load applied to each core of CPUs because of exchange of data between CPU and GPU.

It was confirmed that the processing speed and the memory and CPU usage can be saved as Fig 2.6, Fig 2.7 as a result of optimized method considering the driving characteristics and processing efficiency of the CPU and the GPU.

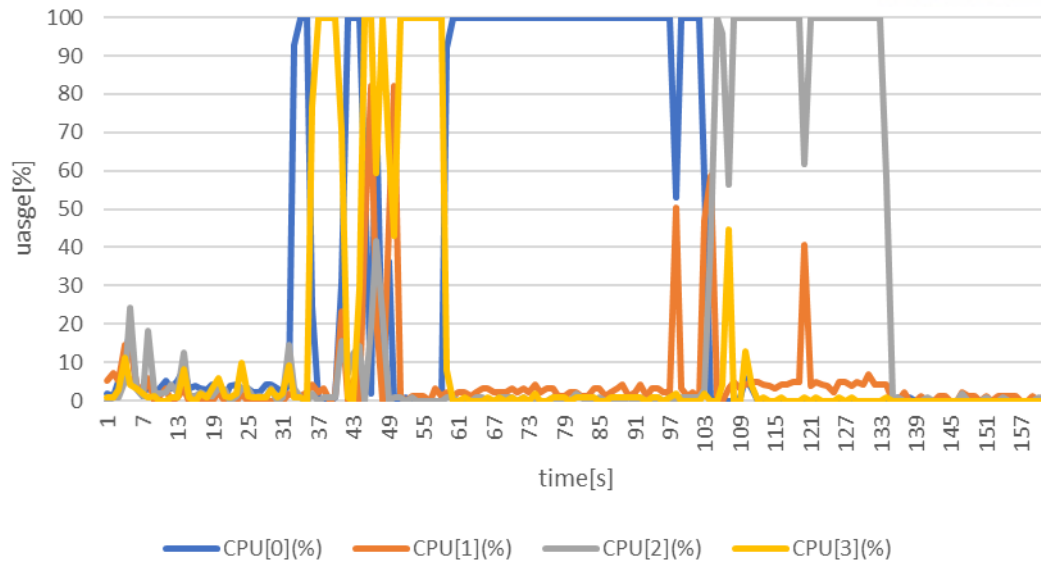


Fig 2.5 Program execution using CPU only

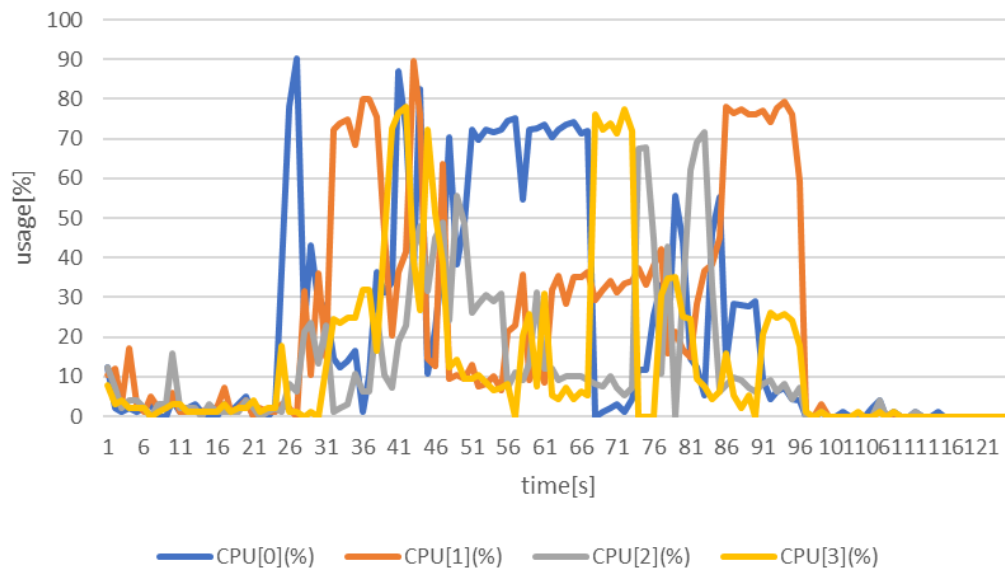


Fig 2.6 Program execution using CPU and GPU

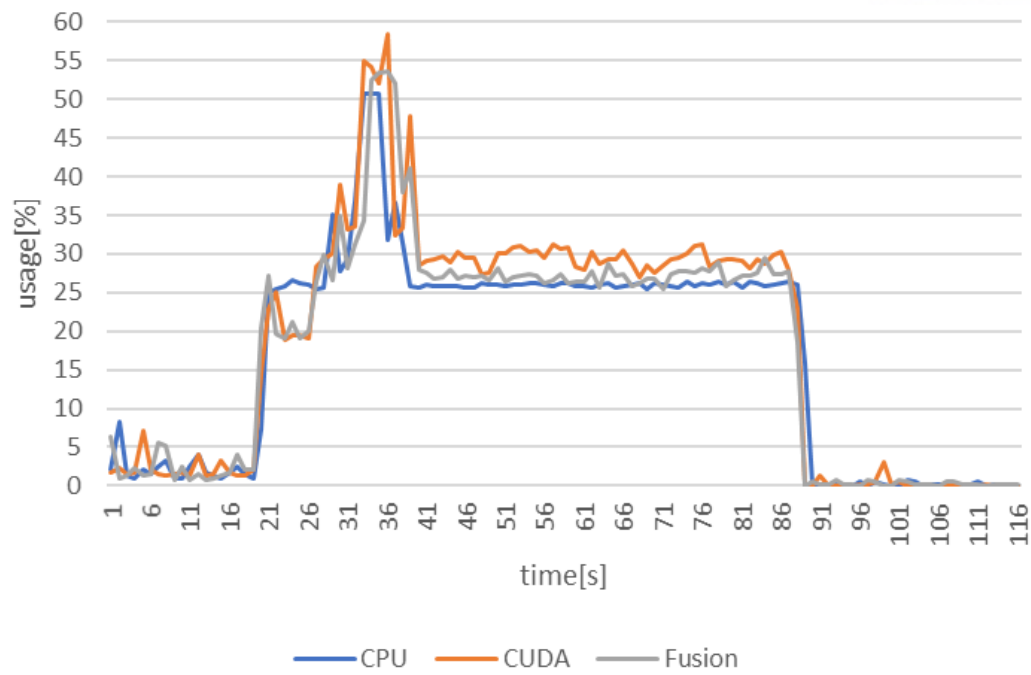


Fig 2.7 Total usage of CPU for each test condition

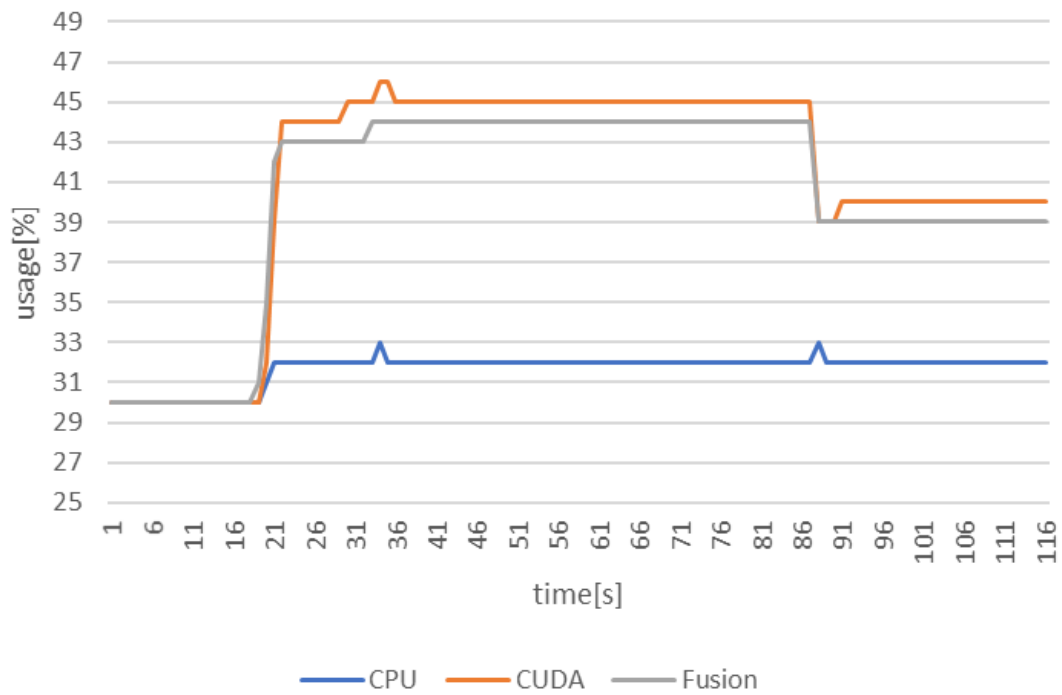


Fig 2.8 Memory usage for each test condition

2.2 Limitation of Overperformance

This approach which is limit FPS to constant minimum speed that does not affect the performing process or mission has been proposed to save surplus resources for other processing. For this test, processing was forced to sleep during as long as the time remains. FPS was adjusted from 13-14 to 10, average of CPU usage is reduced 40% to 30% in maximum running section of Fig 2.9 and Fig 2.10. As a result, computing resource was retained surplus resources by intended limitation of performance, but memory consumption is constant, because limiting the processing speed does not change the amount of data to be processed. It is also necessary to maintain a uniform and stable system, as it also affects the performance of other processes associated with imaging results.

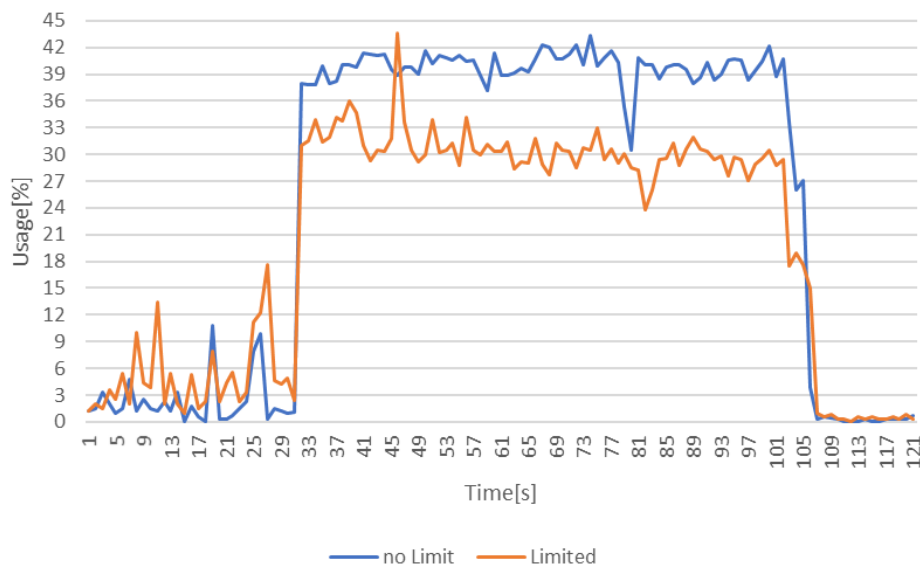


Fig 2.9 Total usage of CPU at limited performance

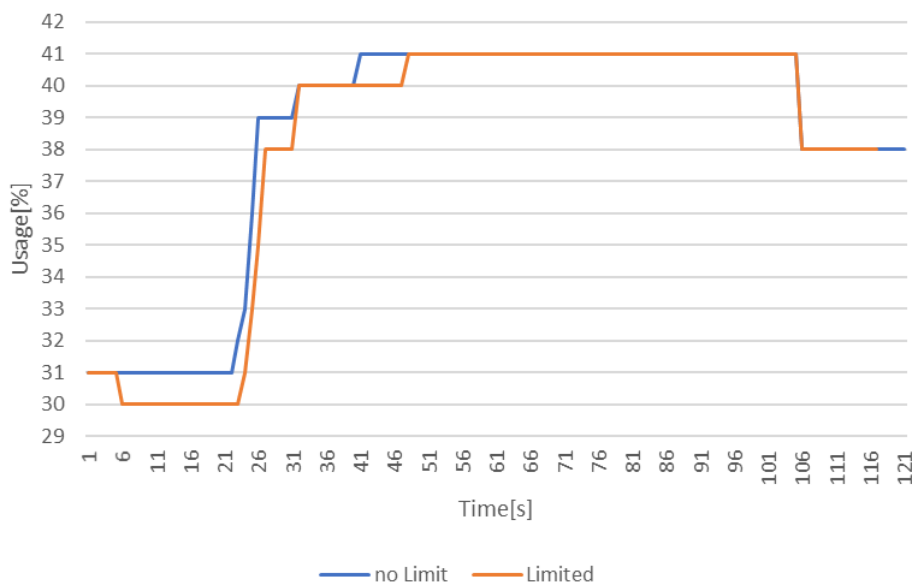


Fig 2.10 Memory usage at limited performance

2.3 Convolutional Neural Network Visualization

In the training session, a large amount of dataset must be stored and processed in memory, it is assumed to be performed in offline main computer considering time and computation efficiency. In other words, online training is not considered, but in the future work, the potential should be considered for the emergence of innovative ideas and advances in hardware. In this work, the parameters learned from the offline computer are transferred to the embedded system.

Deeper layer Neural Network build more complex regression area as shown in Fig 2.11. ‘Deeper’ means an increase in the number of layers. However, it has high risk of facing which vanishing gradient problem. To prevent this, VGG Net proposed using multiple small-sized kernels [15]. It can be basis for determining the number of layers of network, the number and size of the kernel 5x5 and 3x3. Table 2.2 presents designed structure of CNN for this work.

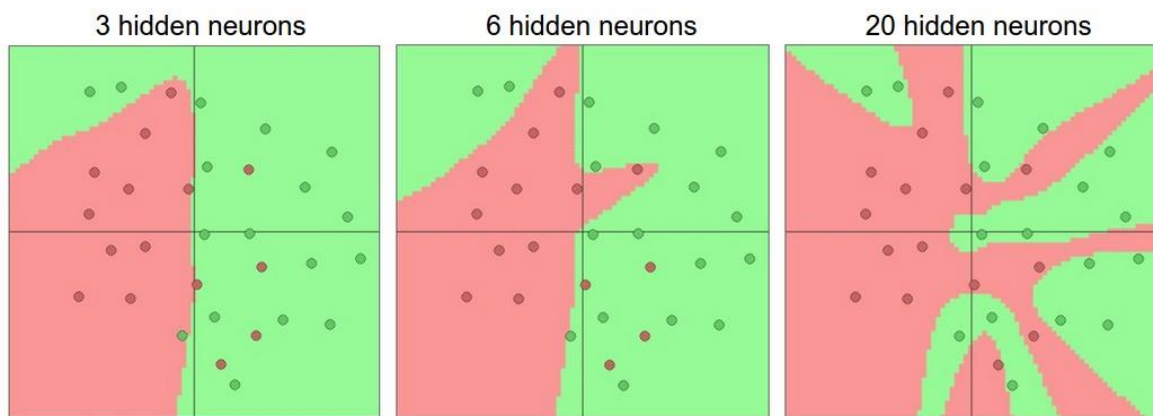


Fig 2.11 Regression area built by multiple layers

Table 2.2 Network design principle

Number of convolution layers	3 and more
Size of kernels	5x5, 3x3
Number of kernels per layer	2~3, less than 5
Number of neurons on fully connected layer	Determined by number of kernel and class

Target image dataset was collected for training as shown Table 2.3. And they were enlarged by affine transform, translation, adding gaussian noise etc. Because enough image dataset is needed to train network at least 1000 patches per each class. And dataset pre-processing helps training more faster and enhancing result of training.

Table 2.3 Dataset specification

Type	Size	Preprocessing	True dataset	False dataset
Crack	64x64	Morphological filtered	1486	1665
Circular target	Various	Canny edge detection	845	338

2.3.1 Training Session Visualization

CNN is too complex to understand mathematical operation, to predict result and to find cause of error or training not going well. Visualization is greatly intuitive technique for analysis, it can be the solution of understanding CNN. In this section, it will be shown how the operation of the network changes with the number of neurons of fully connected layers, and kernel changes by convolution layer. The network specifications used for the test are as shown in Table 2.4 and Fig 2.12.

Table 2.4 Network parameters of trained CNN

		Kernel size	Number of kernels	Pooling size	Activation function
Convolution layer	Layer 1	5x5	3	2x2	ReLU
	Layer 2	3x3	3	2x2	ReLU
	Layer 3	3x3	5	2x2	ReLU
Fully connected layer	Number of neurons	64, 128, 256		-	Sigmoid

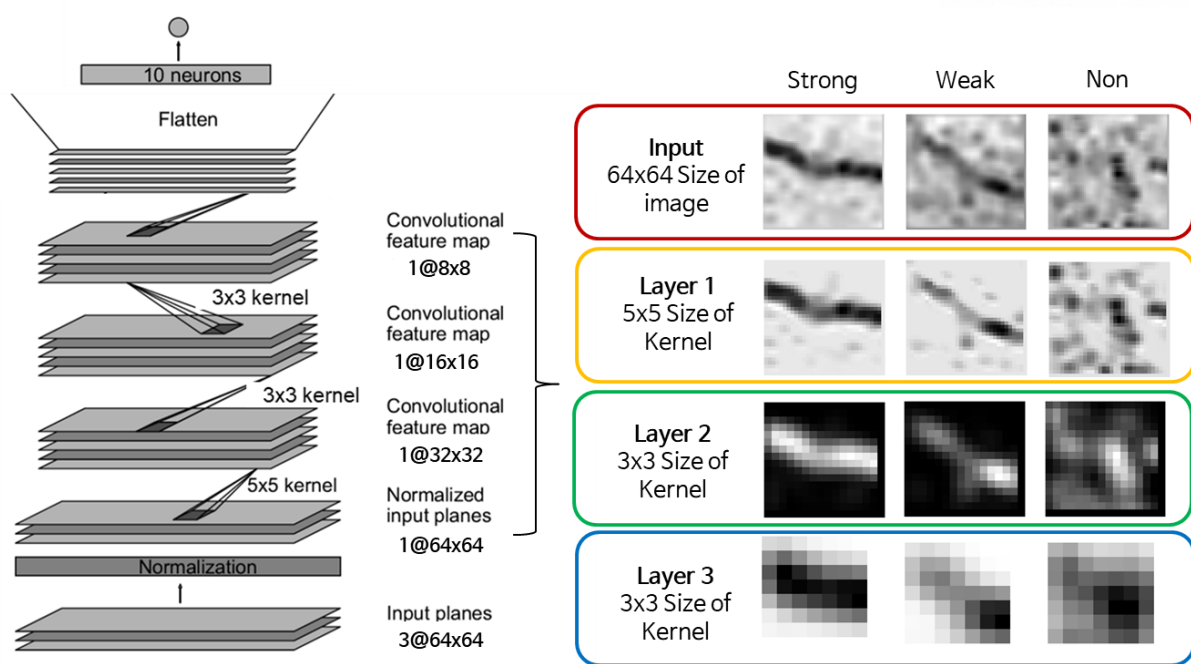


Fig 2.12 Network structure of trained CNN

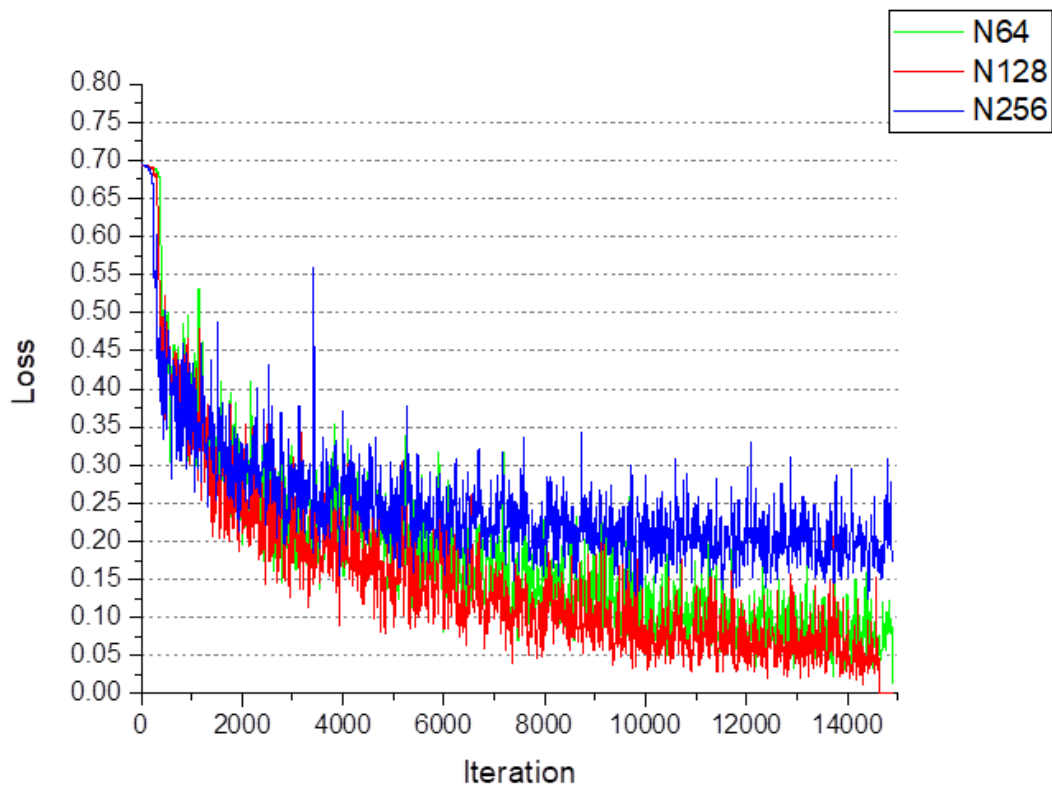


Fig 2.13 Error drop during network training

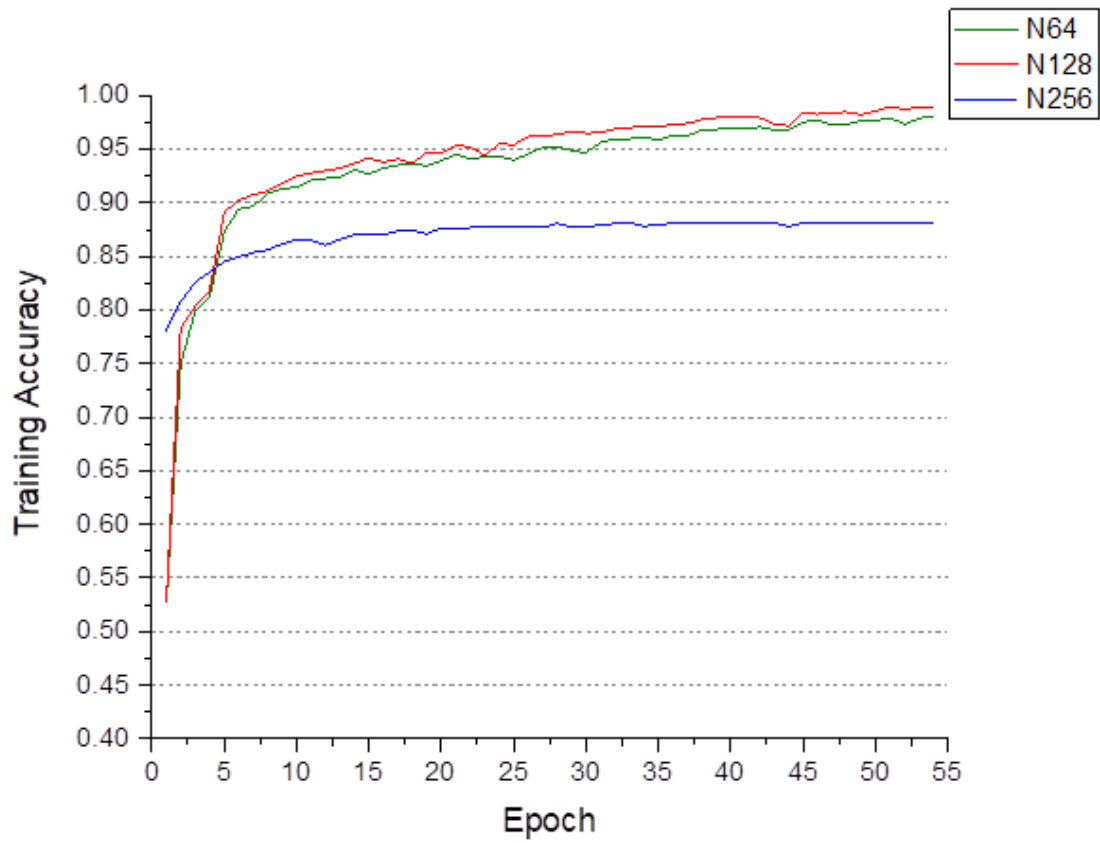


Fig 2.14 Accuracy check for training dataset

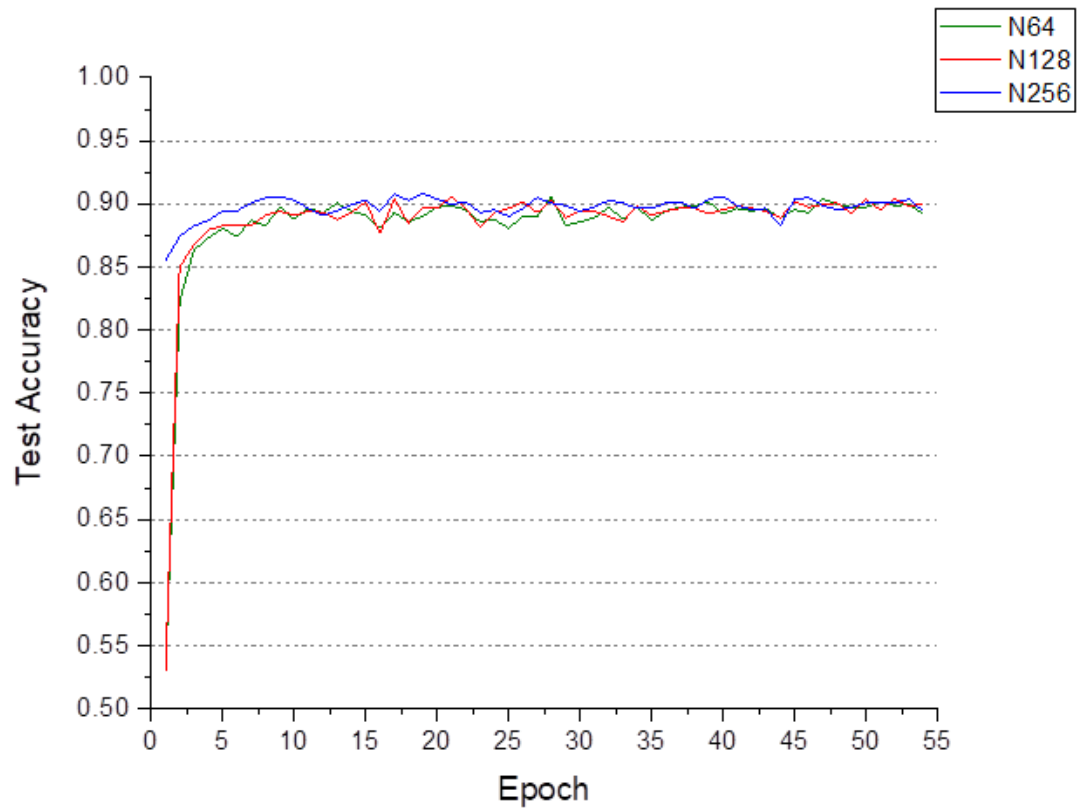


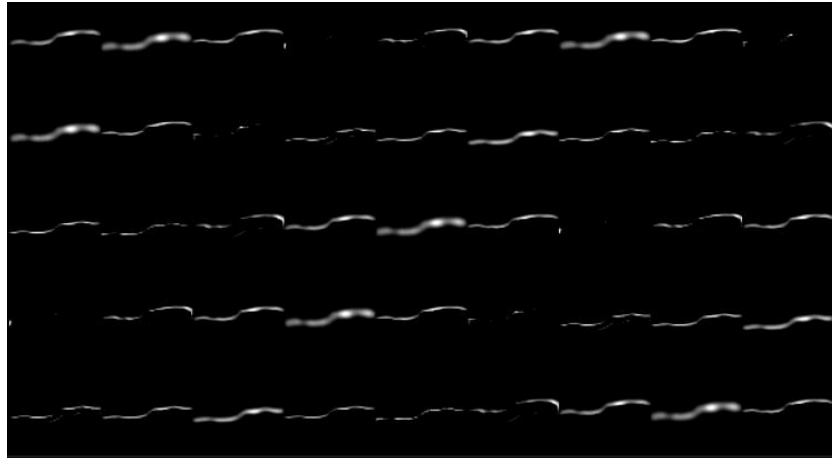
Fig 2.15 Accuracy check for test dataset

$$E_j = \|w_{ij}x - y\|_2^2 = \sum_j^n (w_{ij}x_j - y_j)^2 \quad (2.1)$$

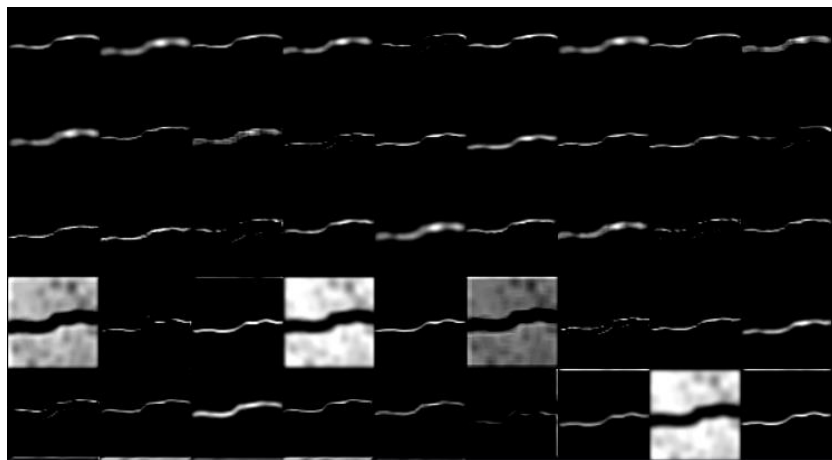
$$\frac{\partial E_j}{\partial w_{ij}} = \frac{\partial E_j}{\partial out_j} \times \frac{\partial out_j}{\partial net_{oi}} \times \frac{\partial net_{oi}}{\partial w_{ij}} \quad (2.2)$$

Result as Fig 2.13 indicates loss of network. Loss is calculated by j^{th} , desired output(y_j) and predicted result($w_{ij}x_j$) multiplying by weights and output of previous layer (2.3). Fig 2.14 and Fig 2.15 present result of dataset accuracy test during training. Test dataset is only for validation, not included parameters updating process. Referring to N64 data in green, it indicates that too low a number of neurons can have a negative effect on the reduction of training loss.

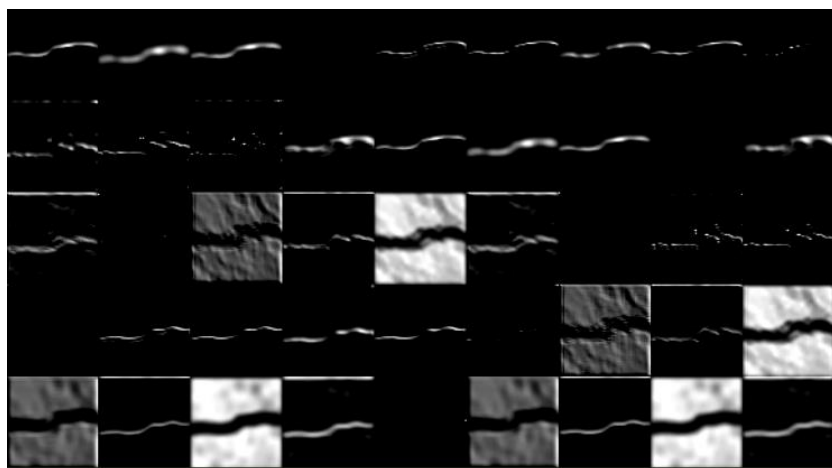
In the graph, when the number of neurons on fully connected layer was cut, the loss was decreased at a slower rate, but as the training progressed, the loss reduced further, and the learning effect was better. The gradients of weights is updated by output error during the training. They are propagated to entire network by the chain rule (2.4). However, error value at the back of the network is lost close to 0 while reaching the front of the network, convolution layer. It is called by vanishing gradient. This phenomenon can be detected in visualized kernels and convolved image with these kernels Fig 2.16



(a) 256 Neurons



(b) 128 Neurons



(c) 64 Neurons

Fig 2.16 Visualization of Convolved image with trained kernels

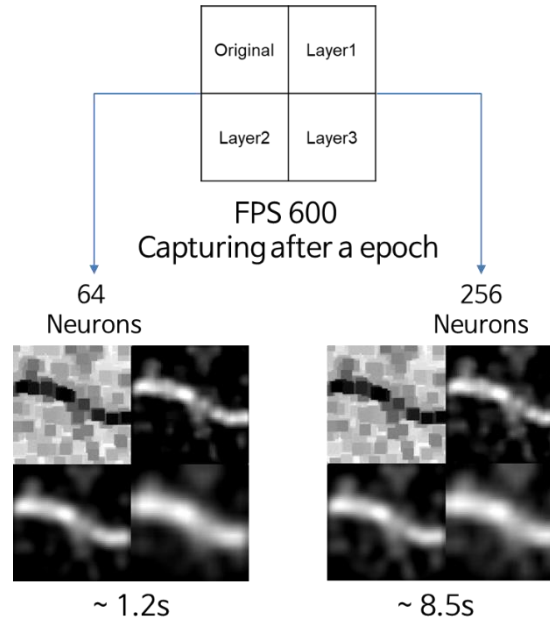


Fig 2.17 Convergence time with number of neurons

From the Fig 2.16, the learning efficiency of the convolution layer is increased, kernels generated much variously by analyzing the reason that the propagation of the error gradient occurred well by reducing the number of neurons in the fully connected layer.

Another evidence supporting this hypothesis is found in the visualization for monitoring the training process. In this process, it can be seen that the convoluted image converges to a certain form, and a network with a larger number of neurons converges more slowly according to result at Fig 2.17.

Also, selective kernel approach can be used refer to visualized kernels and convolved image as Fig 2.18 for efficiency. It will be tested and described at Sec 2.4.

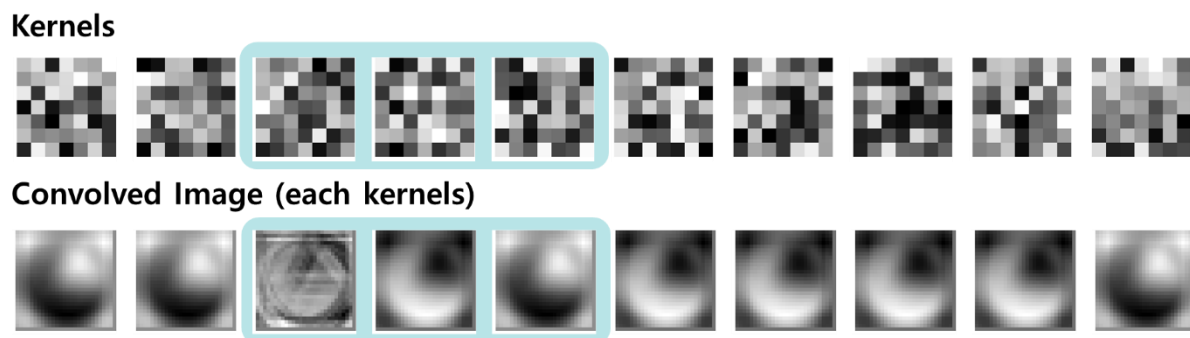


Fig 2.18 Convolution kernel visualization

The benefits of visualization can be summarized as follows

- i. Visualization result helps understanding when training results don't get better
- ii. Intuitively identified CNN
- iii. Basis for training completion with network loss
- iv. Selective selection of convolution kernels

2.3.2 Test Session Visualization

CNN is useful algorithm for object classification and after classification, there are many applications that follow, such as image segmentation, target positioning after classification. However, a fully connected layer that accepts linked image with convolution layer and performs classification with input. It only accepts fixed size inputs, and this process removes the positional information of the signal enhanced by the convolution. To applying on object segmentation and the localization of the detected target requires location information, which causes problems. Fully Convolutional Networks (FCN) considers fully connect layer as the 1x1 convolution operation [16].

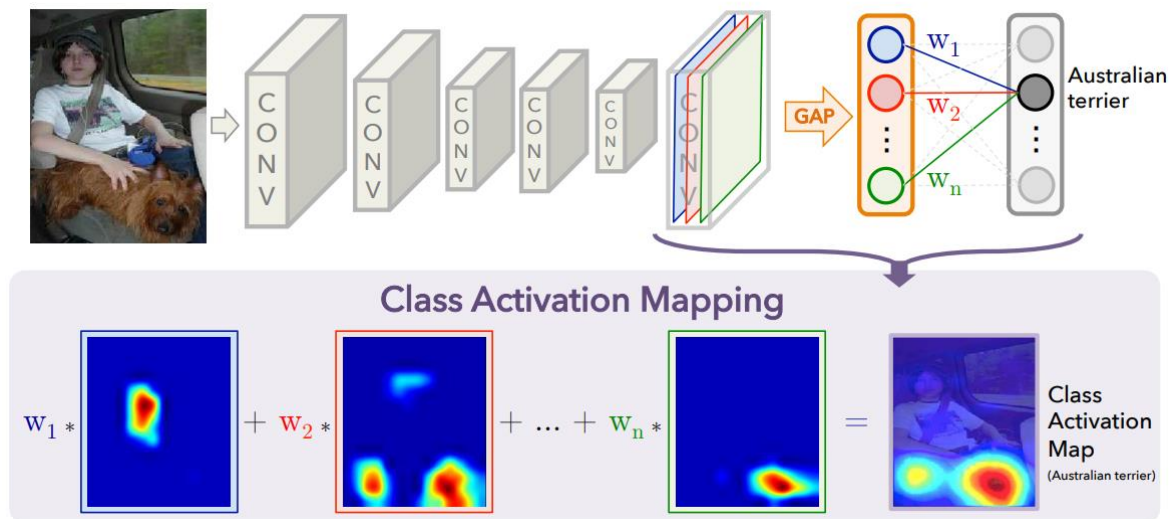


Fig 2.19 Class activation mapping [17]

There are some techniques to set position of object such as sliding window method [8], but that method is taken too long time to process in real-time, So, alternative method should be applied, that is visualization of CNN [17-20]. Activation map visualizes activation of neurons with convolved and Global Average Pooling (GAP) image shown in Fig 2.19 [17].

Each GAP image is connected to the output class with weights that means how important it is in the associated class. Fig 2.20 is processing image by level of neuron activation according to the depth of the network. The deeper network shows more accurate results as show. Target detection and localization can be performed simultaneously or if the target exists in a frame, the activation map demands few nodes that is multiplication between number of convolutional kernels and output class

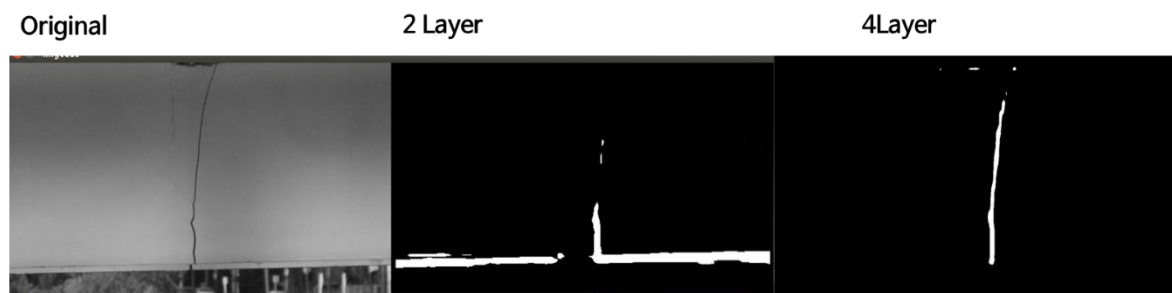


Fig 2.20 Level of activation according to the depth of the network

To compare the result of crack segmentation, image processing technique will be proposed. A number of segmentation methods tried for crack detection but as mentioned in Sec. 1.2, it has threshold decision problem how much value is appropriate and design parameters dominate quality of results. In this work, the biggest feature of people recognizing cracks in a structure is that it has a continuous line with non-direction and dark compared to the brightness of surrounding pixels. Therefore, the algorithm should be constructed to detect cracks based on these characteristics and minimize the effects of the parameters that make up the algorithm.

First of all, Image I is performed maximizing features of crack and noise removal through morphological filters following steps on Table 2.5. After step 1, binarized with Otsu threshold. It makes markers as seed of area segmentation. Watershed is a kind of area segmentation method. With coupling the divided areas to determine whether the existing connection between the two areas is a crack or not. The average value of the pixels in the regions satisfying the condition of (2.1) is compared and the cracks are discriminated by thresholding according to (2.2). All the processing result by step is shown Fig 2.21.

Table 2.5 Procedure of crack detection

Step	Process	Function
1	Morphological filtering, Erode	Maximize features of crack
2	Otsu binarization	Making markers for segmentation
3	Morphological filtering, Erode and Dilate	Noise removal
4	Draw contours	Drawing makers
5	Watershed segmentation	Dividing cracked area
6	Area indexing and binding	Finding contact contour
7	Thresholding (2.1), (2.2)	Crack discriminating

$$p_{ij} \in I_{area}, p_{kl} \in C_{crack} \quad (2.3)$$

$$\left(\frac{\sum p_{ij}}{n_{area}} - \frac{\sum p_{kl}}{n_{contour}} \right) < threshold \quad (2.4)$$

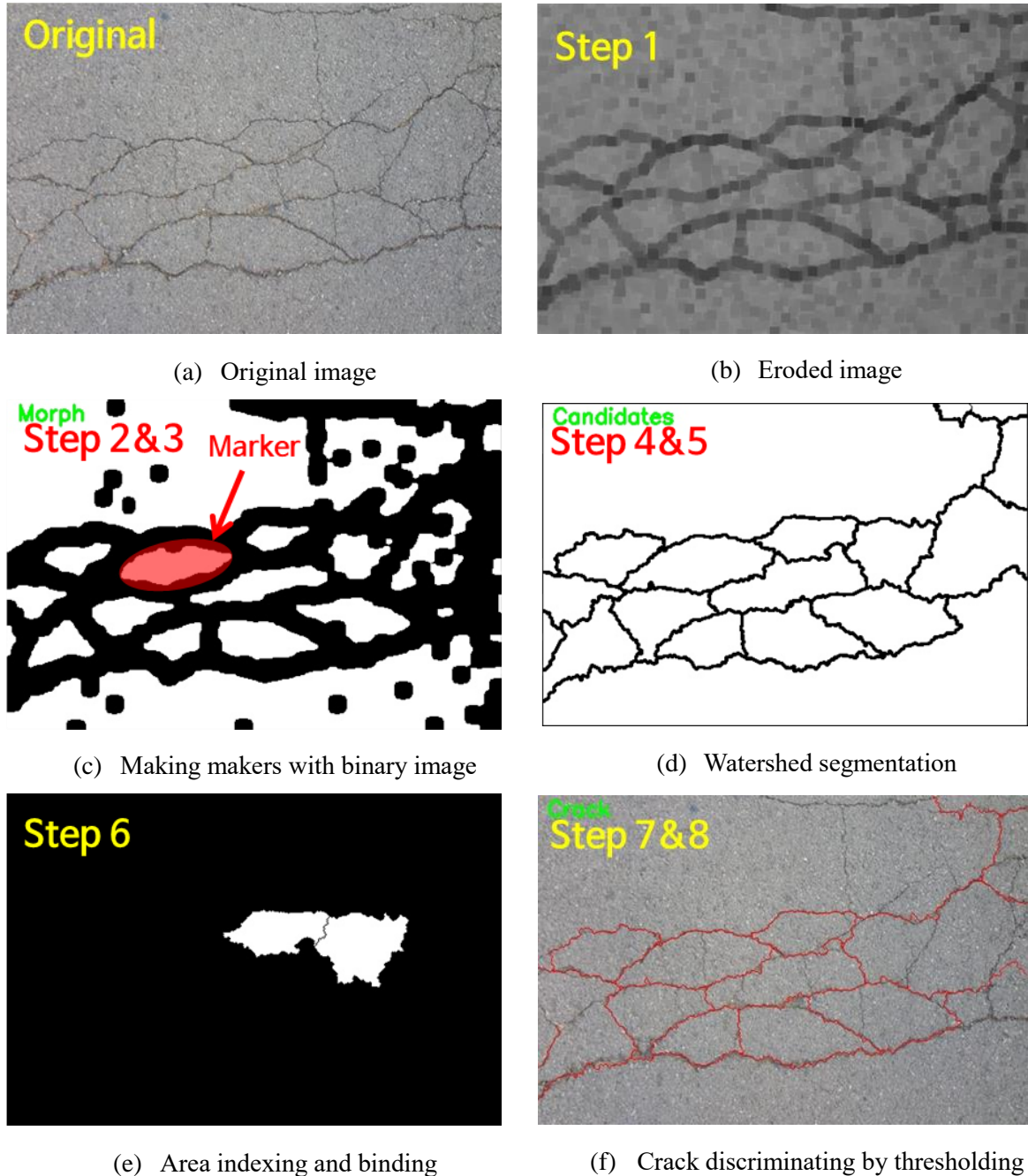
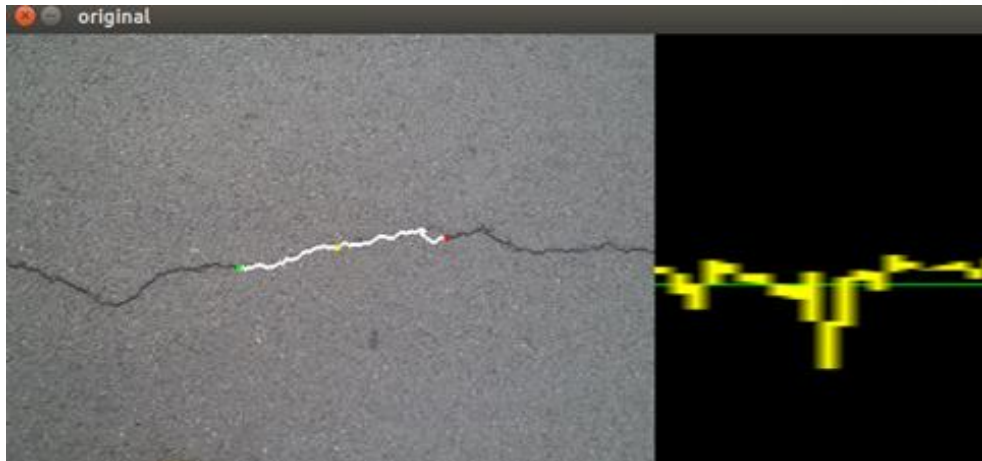
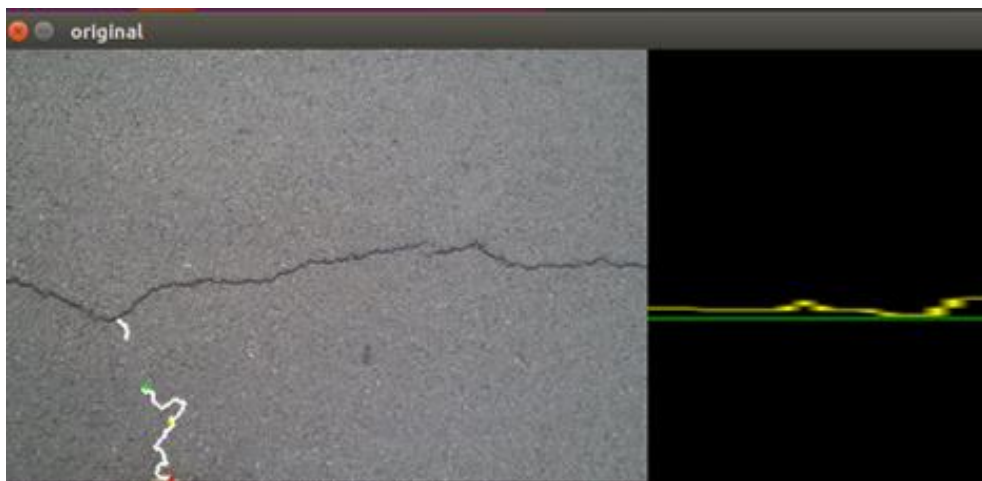


Fig 2.21 Crack detection process

However, there are still limitations to using the threshold. Fig 2.22 shows the results of a method that minimizes the impact from the threshold. When the values of the pixels that are orthogonal to the connecting line which is segmented are plotted as a graph, if the variance of the passing through contour is large, it has high possibility of crack and if the graph shows uniform pixel values, it may not be the crack. The yellow line indicates pixel value by perpendicular line of crack candidate contour, the green line is the mean value of whole contour.



(a) Crack



(b) Not crack

Fig 2.22 Discrimination crack

2.4 Region proposal

Resolution refers to the quality of images and captures detailed characteristics. Even small objects can be detected, so the detection performance is directly proportional to the resolution, but an appropriate compromise between performance and processing speed is needed because the amount of data needed to calculate increases dramatically as the resolution increases. Thus, the Region of interest(ROI) approach extracts and processes only a portion of the image, which can be less affected by resolution, thereby increasing the efficiency of the algorithm.

If the algorithm selects a candidate area to detect a target and examine only that area, the processing speed and accuracy will be improved. ROI selection can be made using the characteristics of image data such as color, material, shape, etc. So, region proposal based on pre-processing by image processing. ROI is important circular target detection. For this case, ROI can be choose using Hough circle transform of Affine moments invariants(AMIs). The sample algorithm was built as an example, it consists of image processing part for ROI selection and feedforward network to classify ROI from result of image processing as Fig 2.23. Network condition and training condition were described in Table 2.6 and Table 2.7.

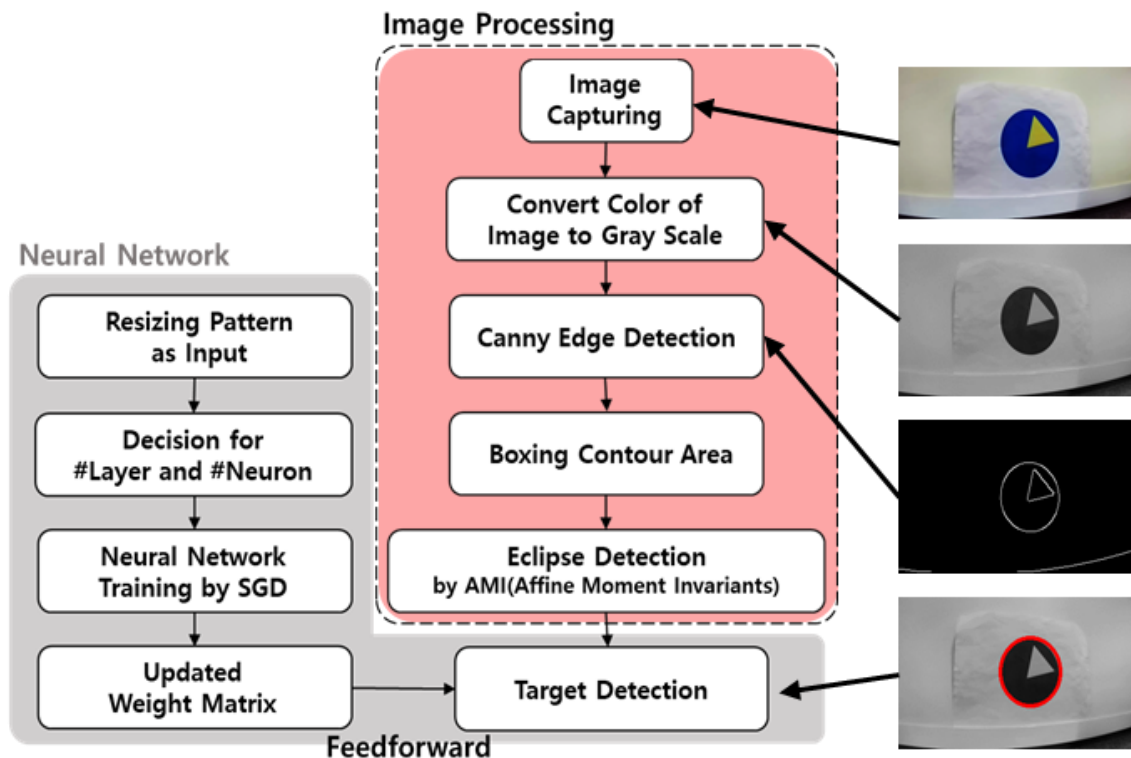


Fig 2.23 ROI proposal as input of CNN

Table 2.6 Network condition

Input (=Image Size)	Convolution Layer	Kernel	Kernel Size	Hidden Layer	Neuron	Output
784 (=28x28)	1	3,4,10	7x7	1	64	2

Table 2.7 Training condition

Training dataset	Validation dataset	Learning rate	Momentum	Training accuracy goal
1276	548	0.001	0.9	98%

The overall algorithm operation speed was limited to FPS 10, and the processing speed was verified by measuring the time taken per feedforward in Table 2.8. Delays do not occur even when there are more than 20 input patterns that have gone through image processing.

It was confirmed that continuous detection of deformation such as rotation and tilt is possible and that similar patterns can be distinguished through the neural network as shown in Fig 2.24 and Fig 2.25, but incorrect detection occurred depending on the size or shape of the detected pattern in Fig 2.26. CNN may also not be able to learn well due to incomplete elements such as noise, resolution, unobtrusive features in the filming environment, thereby reducing accuracy.

Also, case 3 and case 4 were tested with selective kernels, it shows similar performance as much as using all kernels trained in Fig 2.27, but also lower hardware resource consumption used as shown in Fig 2.28.

Table 2.8 Processing time on real-time CNN

Case	Kernel	Convolution layer[ms]	Neuron	Fully connected layer[ms]
(1)	-	-	-	-
(2)	10	3.5~4.08	64	0.96~1.3
(3)	4	1.38~2.0	64	0.24~0.53
(4)	3	1.06~1.23	64	0.18~0.35

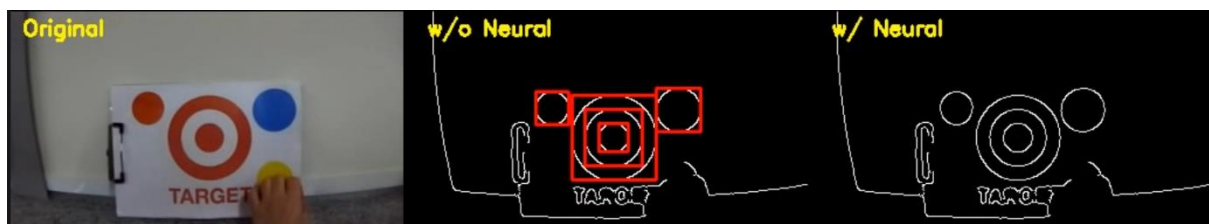


Fig 2.24 Test result(1) filtering

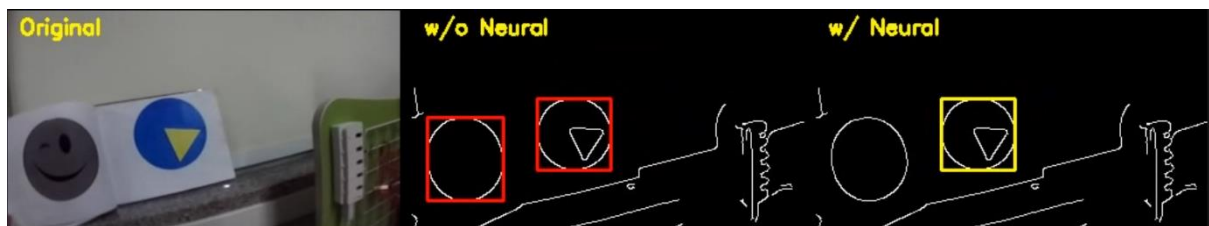


Fig 2.25 Test result(2) filtering and detection

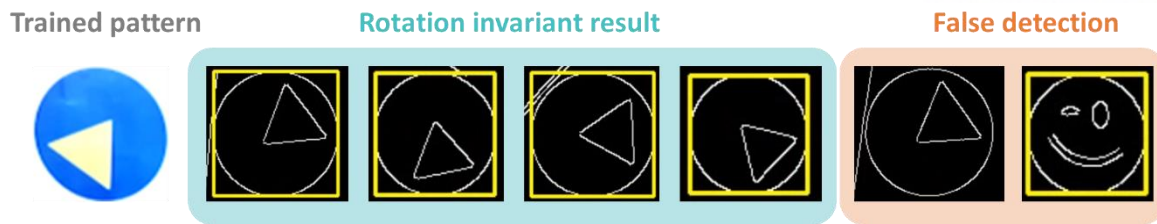


Fig 2.26 Test result(3) rotation invariant and fault result

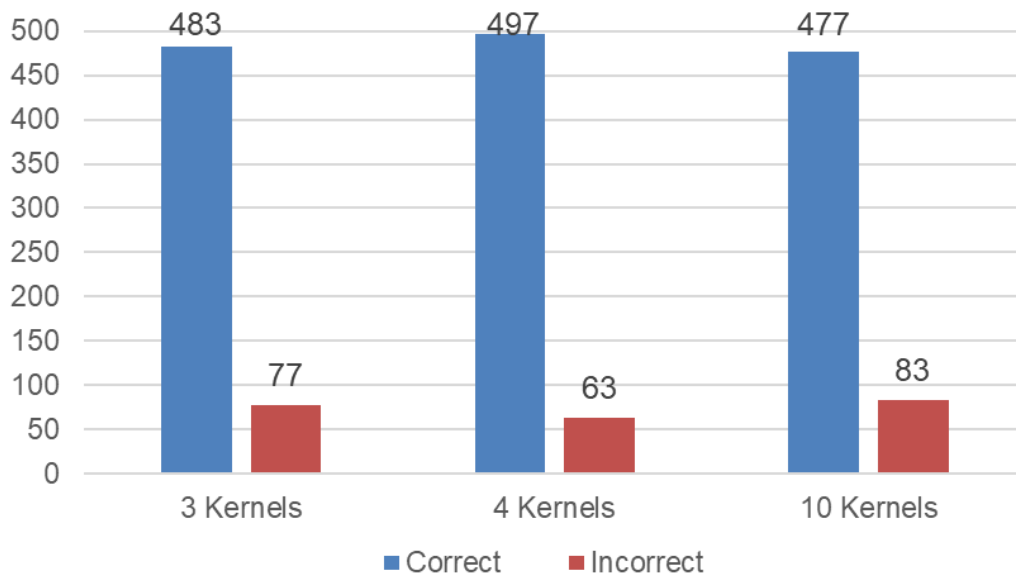


Fig 2.27 Result of proposed algorithm with various kernels

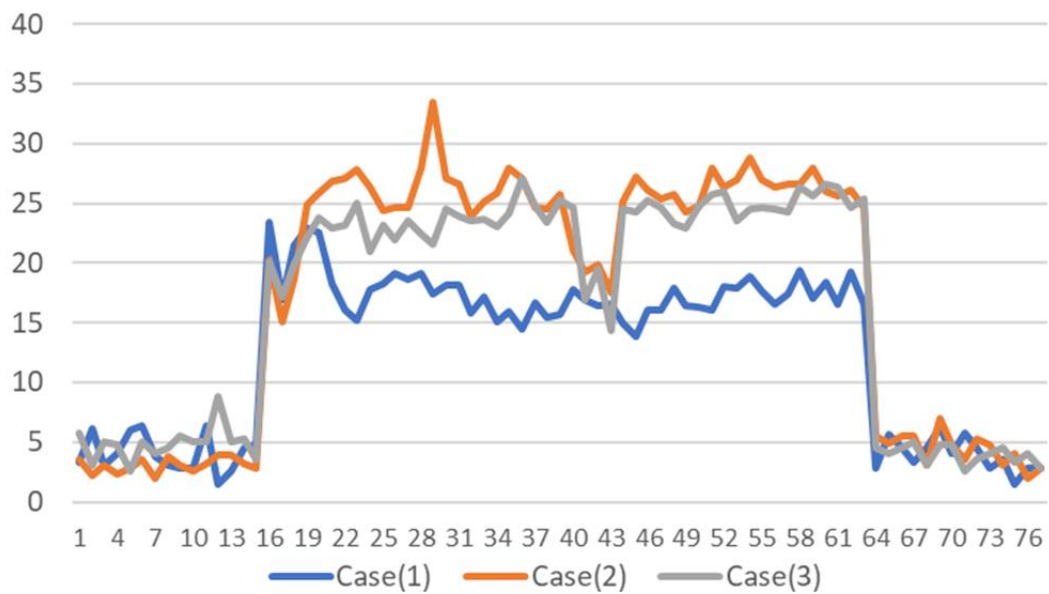


Fig 2.28 CPU total usage [%]

III. IMPLEMENTATION OF UAVS

NVIDIA Jetson TX1 as computing board of Fig 3.1 was used for test, also ZED stereo camera of Fig 3.2 was taken as image sensor. It provides own API that can be installed for development. Through the API, the advantage is that stereo image matching is performed by an internal processor, which enables rapid use of depth information. Pixhawk is a kind of microcontroller used as flight controller which is communicated by Mavlink message packet. It has various sensors for attitude estimation such as magnetometer, accelerometer and barometer. It is shown in Fig 3.3. All the specification of product was arranged in the Table 3.1-3. A quadrotor drone of Fig 3.4 was set as test platform with computing board, image sensor and flight controller. The optical motion capture module of Fig 3.5 was used for record ground truth data such as attitude and position.



Fig. 3.1 Jetson TX1

Table 3.1 Jetson TX1 Specification

CPU	ARM Cortex-A57 (quad-core) @ 1.73GHz
GPU	256-core NVIDIA Maxwell @ 998MHz
RAM	4GB 64-bit LPDDR4 @ 1600MHz
OS	Ubuntu 16.04
Dimension	50mm x 87mm x 40mm



Fig. 3.2 ZED Stereo camera

Table 3.2 ZED Stereo Camera Specification

	Value	Notes
Resolution	VGA to 2.2K	FPS 100 to 15
Depth range	0.5~20m	32bit
FOV	90°(H) x 60°(W)	-
Power	USB(5V)	-
API Support	O	-



Fig. 3.3 Pixhawk

Table 3.3 Pixhawk Specification

Processor	ARM Cortex M4 256 KB RAM
Sensors	MPU6000 as main accel and gyro ST Micro 16-bit gyroscope ST Micro 14-bit accelerometer/compass (magnetometer) MEAS barometer
Power	7V
Interface	UART, I2C, ADC input, Satellite input, PPM ..



Fig. 3.4 Test platform setup



Fig. 3.5 Optical motion capture system

3.1 Setpoint Generation

All the setpoints were generated for velocity control in body frame as result of image processing for control, because camera is mounted on frame of drone. The center image of Fig 3.6 has noisy data generated because some pixels are lost during the matching process to get depth data.

In order to more accurately measure the distance to the object, we used two methods of ignoring or correcting the noise. The first is to use a morphology filter that can close the internal hole by applying erode after dilation. Then, the empty data is then corrected by replacing it with the surrounding one. Another method uses histogram with inside of object contour area and the most numerous values are chosen to decide depth of object. This result shown the right image of Fig 3.6.

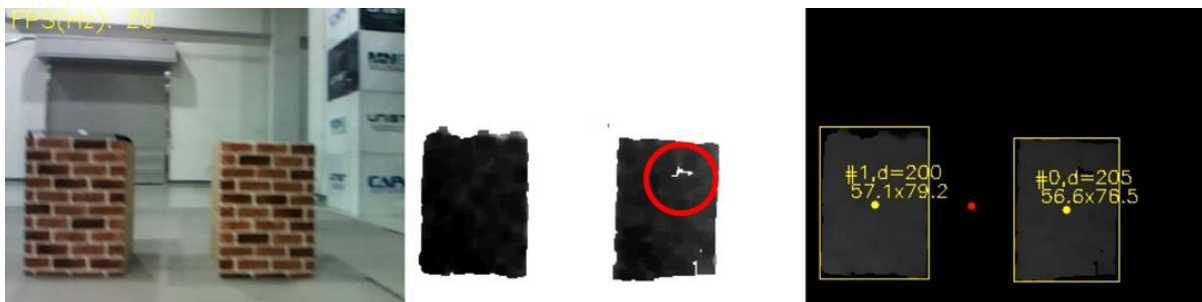


Fig. 3.6 Depth data correction

When two different objects appear to overlap in the image plane, they can be separated using depth map and setpoints can be created so that they are viewed from the front. Fig 3.7 shows that situation

with conceptual drawing. d_{AB} indicates distance between object A and B in frontal view and x_{AB} is the distance between two objects viewed from outside the front for the image plane. With this information, delta yaw from current yaw state and velocity in body frame can be calculated refer to Fig 3.8 and (3.1-3) until delta yaw is zero. w_z can be set considering safety and objective of mission.

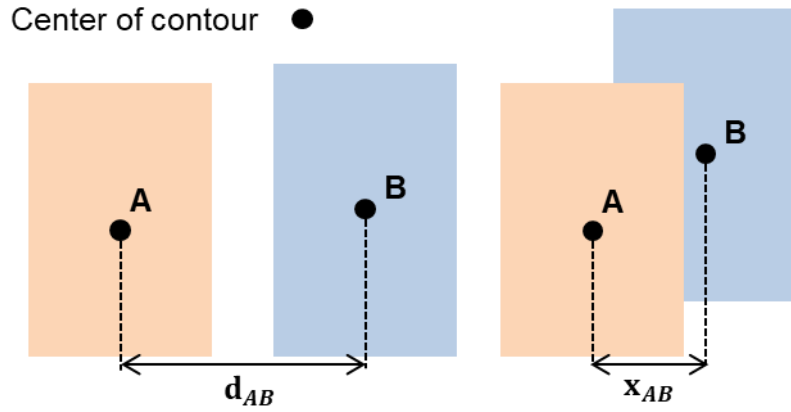


Fig. 3.7 Visual change according to camera position

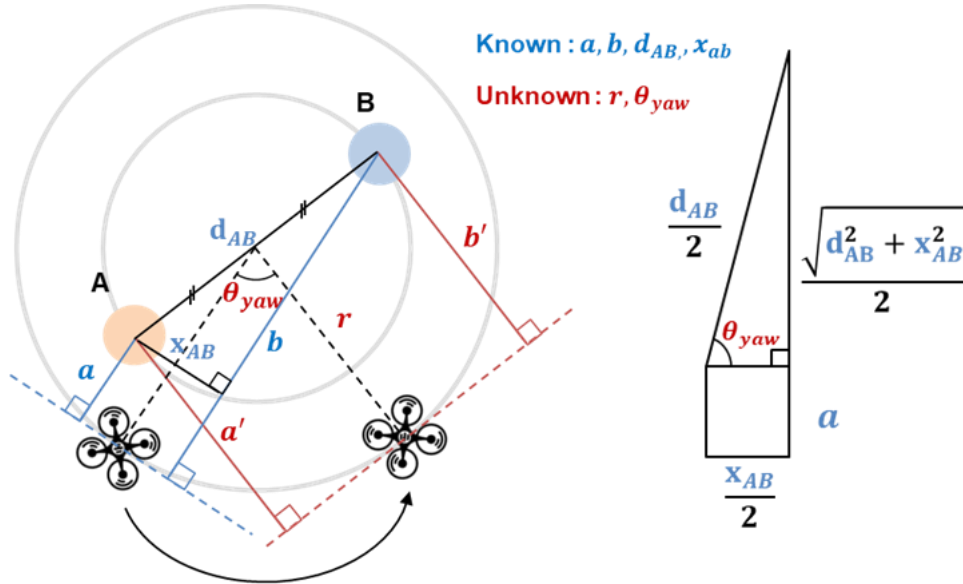


Fig. 3.8 Setpoint calculation using depth data

$$\Delta\theta_{yaw} = \cos^{-1}\left(\frac{x_{AB}}{d_{AB}}\right) \quad (3.1)$$

$$r = a + \frac{\sqrt{d_{AB}^2 + x_{AB}^2}}{2} \quad (3.2)$$

$$v_{by} = r \cdot w_z \quad (3.3)$$

Fig 3.9 shows change of yaw and the resulting delta yaw. Finally, with the drone facing the object head-on, the yaw state from the local frame is placed to zero, so the result should be zeroed when the yaw state and delta yaw are combined, but the error value is much fluctuated, because of noise and image shake from moving of drone. It should be estimated and corrected to improve accuracy.

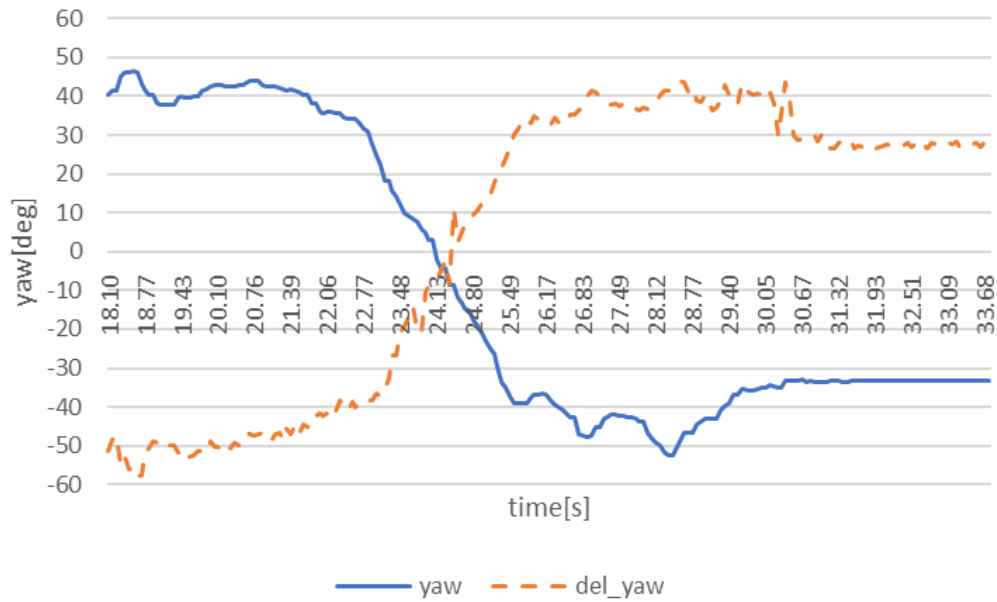


Fig. 3.9 Yaw state data and setpoint

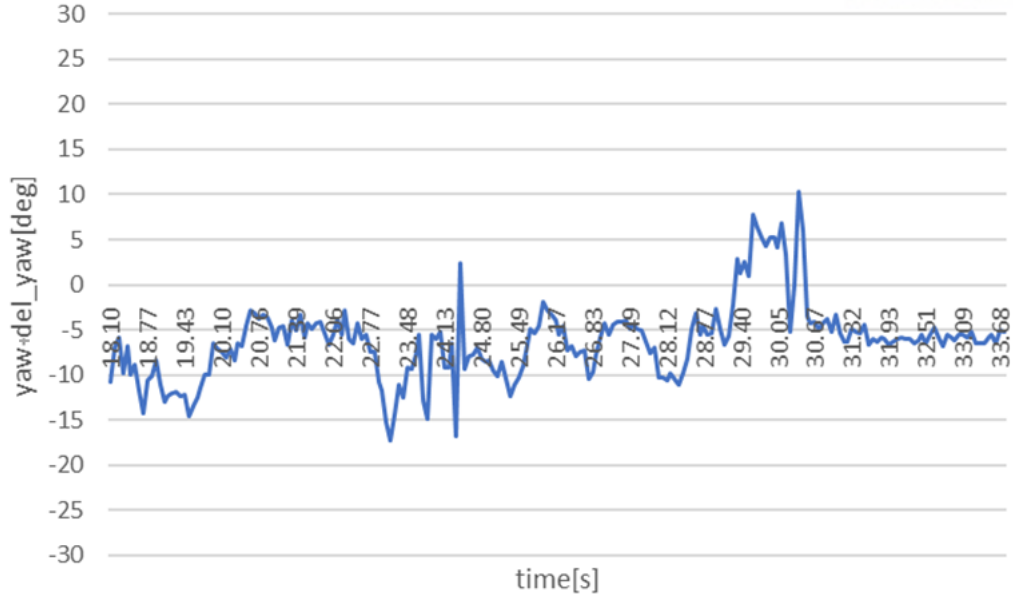


Fig. 3.10 Error of generated setpoint

3.2 Collision Potential Area

With depth information by stereo vision, collision potential area can be drawn potential collision area in image plane. Drone size in the real world can be converted to size in image plane by the depth data and a similar pyramid. And the units were also converted from meters to pixels by (3.4) and (3.5) and the process was plotted in Fig 3.11. So, if result of (3.6) is not empty image, there is a collision potential area indicated with red box meaning of warning like Fig 3.12 and Fig 3.13.

Multiple green boxes in Fig 3.13 shows the size of the drone in a rectangular box along the distance from drone. In the left image of Fig 3.13, if a collision potential area is detected, it has been implemented so that the driving direction can be moved to other areas around it to avoid a collision risk.

$$FOV_x[m] = 2 \times \tan\left(\frac{FOV_x[rad]}{2}\right) \times d[m] \quad (3.4)$$

$$W_{drone}[px] = W_{drone}[m] \times \frac{resolution_x[px]}{FOV_x[m]} \quad (3.5)$$

$$A_{drone}^d \cap A_{object}^d \quad (3.6)$$

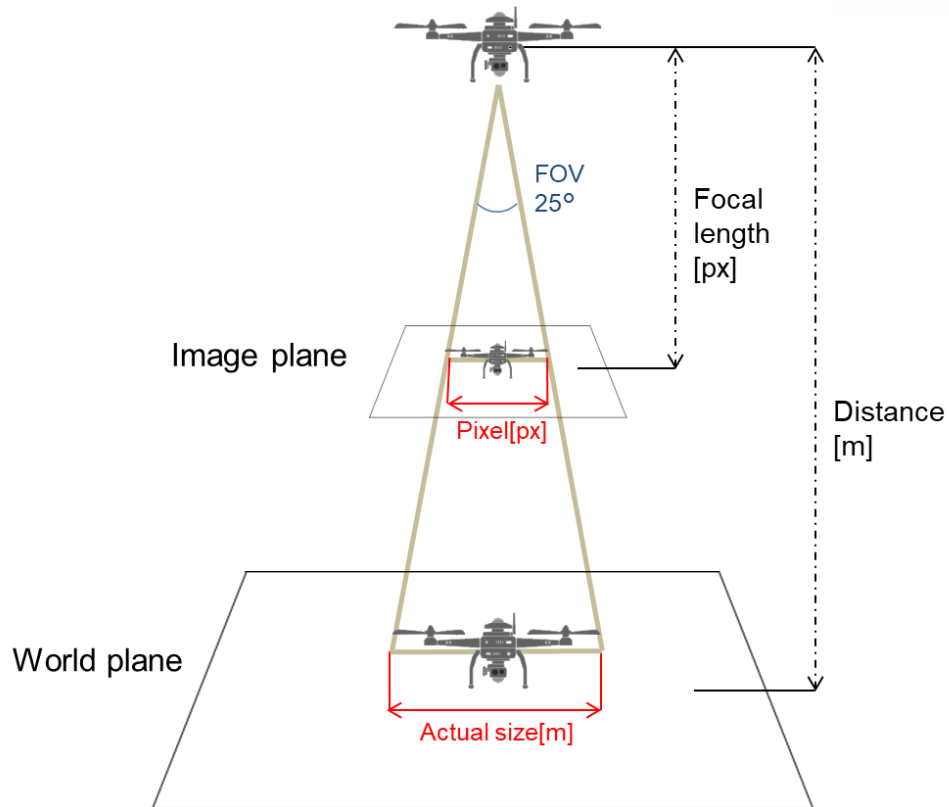


Fig. 3.11 Drone size in pixel by depth



Fig. 3.12 Processed image with collision potential area and object information

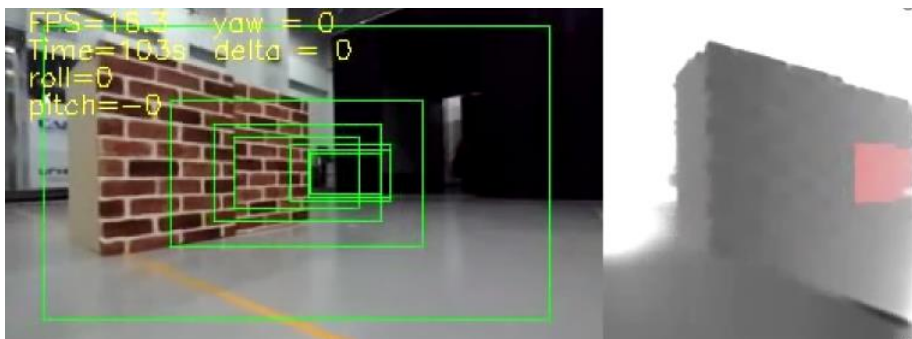


Fig. 3.13 Processed image with collision potential area and avoiding box

3.3 Odometry Assistant using Stationary Landmark

Visual odometry is one of the useful technique for positioning of UAV on local frame such as Simultaneous localization and mapping(SLAM), data fusion with inertial measurement unit(IMU) sensor and homography. Using the homography matrix, it is possible to estimate the relative position and attitude of the camera in the local frame, but the process of finding and matching the features and calculation the multi-dimensional matrix is complicated and consumes a large amount of computation. Therefore, in this section, attitude information represented by Euler angles from the depth map and IMU is used to create the odometry through the relative position between the landmark object and the drone.

Euler angles represent the rotation of a body from the world frame. They are defined as three rotations, phi (ϕ) for x-axis rotation, theta (θ) for y-axis rotation, and psi (ψ) for z-axis rotation, relative to the three major axes of the coordinate frame. So the body frame and the world frame can be converted to each other with rotation matrix in Fig 3.14 and (3.7). The rotation order is Z-Y-X when going from world frame to body frame.

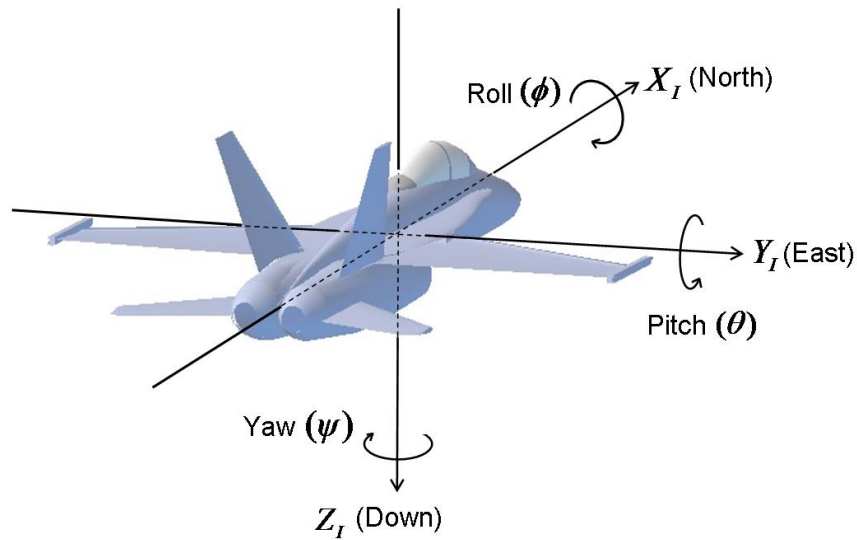


Fig. 3.14 UAV's body NED frame

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = R_w^b(\phi, \theta, \psi) \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = R_\phi R_\theta R_\psi \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \quad (3.7)$$

A multi-rotor drone changes position by inducing a change in thrust direction using thrust and postural control, the posture of the camera mounted on the body, unless the gimbal is used, changes together. As the attitude of the camera changes, the image containing the landmark is also rotated and moved. Therefore, attitude data can be transferred from the flight controller and used for keeping erected image based on the local frame considering the order of rotation. The erected images from roll and pitch are shown in Fig 3.15 and Fig 3.16. Fig 3.17 shows the change in relative position after correction processing.

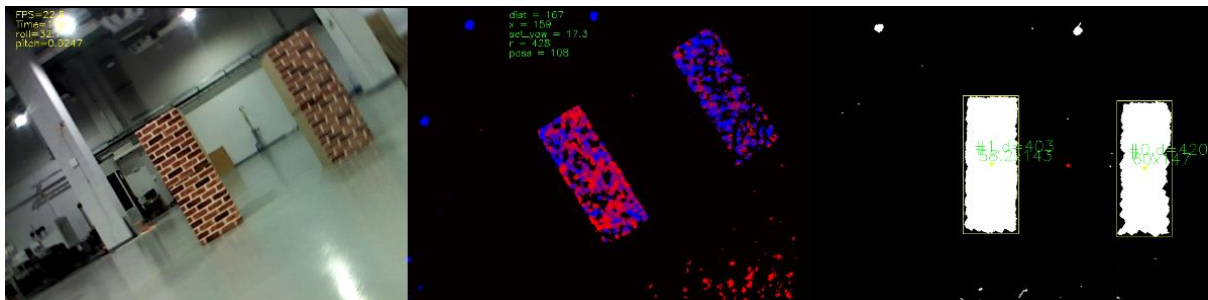
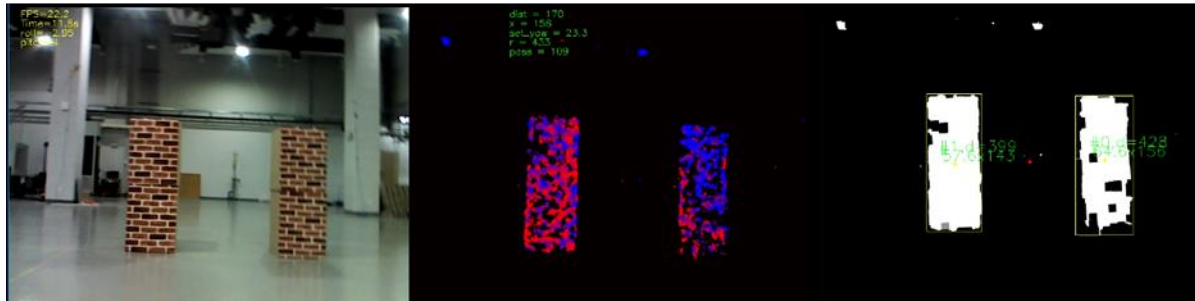


Fig. 3.15 Image adjustment for rotation by x-axis

Pitch = -4 deg



Pitch = 16.1 deg

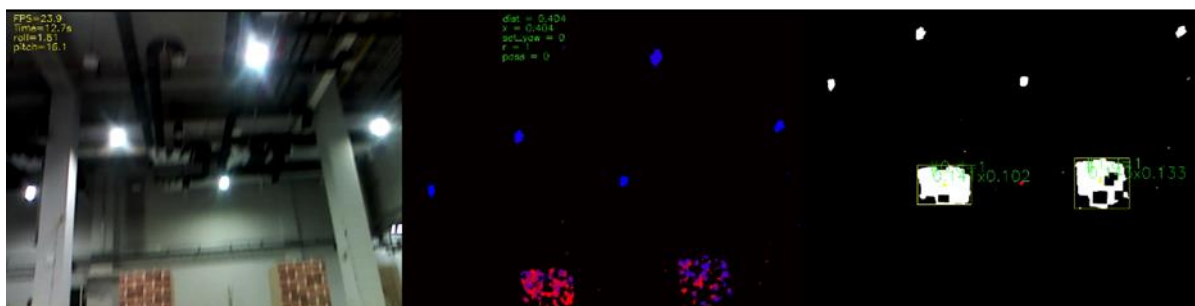


Fig. 3.16 Image adjustment for rotation by y-axis

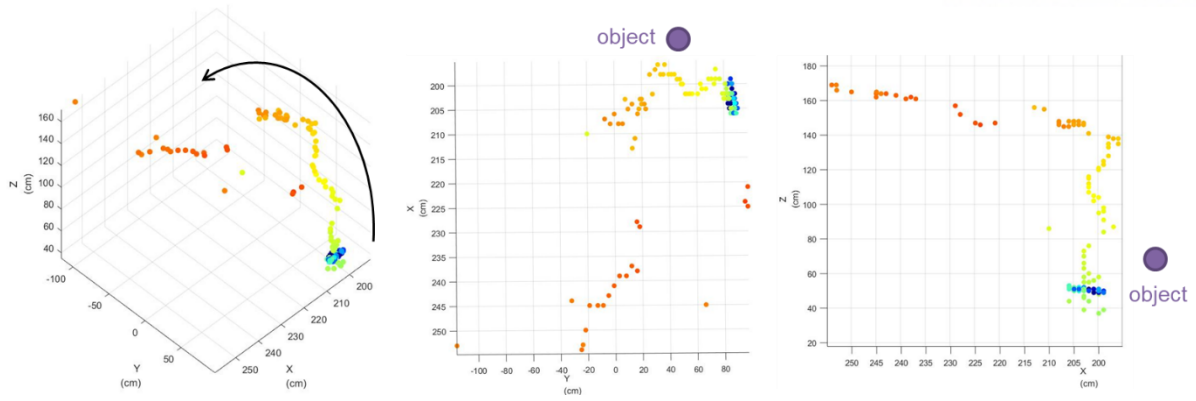


Fig. 3.17 Odometry assistant with object as landmark

IV. CONCLUSION

This work presents a computing optimization approach for embedded system specially on aerial vehicles and micro robots which should keep limited hardware. All embedded systems should be efficient and managed through hardware resource monitoring.

Previously, the method for visual recognition of cracks was proposed to reduce the dominant structure of the parameters of the imaging algorithm, and crack detection and localization was performed through a neuron activation map. This allowed us to respond to the processing of images in a more diverse environment by minimizing parameter design elements.

Also, in order to minimize the computing load of the algorithms driven by the embedded vision system, GPGPU technique was used to reduce the usage of CPU and increase processing speed through parallel operations, and memory management provides a method that can be used at a level that does not stop the system. In addition, ROI selection was performed through pre-processing to reduce the throughput of CNN, and the optimal network structure was selected through visualization at the Training stage, and the kernel was selected and simplified. Finally, valid results were obtained using methods such as securing surplus resources through speed limit.

As discussed in Sec 2.4, the range of embedded visual systems that can be operated on the UAV is various and requires planning and monitoring to consider computing resources and maximize efficiency throughout the entire process. Especially for optimized application of CNN, there are still so many kinds of future work. In order to guarantee uniform image processing performance in an embedded system operating under various conditions, it is necessary to study the optimal parameters through neural network so that contrast, exposure, and sharpness of images can be adjusted. Also, research on transplanting previously validated CNN algorithms into embedded environments will also be needed.

REFERENCES

- [1] E. Pastor, C. Barrado, P. Royo, E. Santamaria, J. Lopez, and E. Salami, “Architecture for a helicopter-based unmanned aerial systems wildfire surveillance system,” *Geocarto Int.*, vol. 26, no. 2, pp. 113–131, 2011.
- [2] P. G. Martin *et al.*, “3D unmanned aerial vehicle radiation mapping for assessing contaminant distribution and mobility,” *Int. J. Appl. Earth Obs. Geoinf.*, vol. 52, pp. 12–19, 2016
- [3] E. Pastor, J. Lopez, and P. Royo, “UAV Payload and Mission Control Hardware/Software Architecture,” *IEEE Aerosp. Electron. Syst. Mag.*, vol. 22, no. 6, pp. 3–8, 2007
- [4] C.-C. J. Kuo, “Understanding Convolutional Neural Networks with A Mathematical Model,” *J. Vis. Commun. Image Represent.*, vol. 41, pp. 406–413, 2016.
- [5] S. Saripalli, J. F. Montgomery, and G. S. Sukhatme, “Vision-based Autonomous Landing of an Unmanned Aerial Vehicle,” in *IEEE International Conference on Robotics and Automation*, 2002, vol. 3, pp. 2799–2804.
- [6] M. Bojarski *et al.*, “End to End Learning for Self-Driving Cars,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [7] R. Medina, J. Llamas, J. Gómez-García-Bermejo, E. Zalama, and M. Segarra, “Crack Detection in Concrete Tunnels Using a Gabor Filter Invariant to Rotation,” *Sensors (Switzerland)*, vol. 17, no. 7, p. 1670, 2017.
- [8] S. Yokoyama and T. Matsumoto, “Development of an automatic detector of cracks in concrete using machine learning,” *Procedia Eng.*, vol. 171, pp. 1250–1255, 2017
- [9] W. Zhang, Z. Zhang, D. Qi, and Y. Liu, “Automatic Crack Detection and Classification Method for Subway Tunnel Safety Monitoring,” *Sensors(Switzerland)*, vol. 14, no. 10, pp. 19307–19328, 2014.
- [10] M. Eisenbach, R. Stricker, D. Seichter, A. Vorndran, T. Wengelfeld, and H.-M. Gross, “Speeding up Deep Neural Networks on the Jetson TX1,” in *International Joint Conference on Neural Networks (IJCNN)*, 2017.
- [11] W. Zhang, D. Zhao, L. Xu, Z. Li, W. Gong, and J. Zhou, “Distributed Embedded Deep Learning based Real-time Video Processing,” in *IEEE International Conference on Systems, Man, and*

Cybernetics, 2016.

- [12] M. Valdenegro-Toro, “Real-time convolutional networks for sonar image classification in low-power embedded systems,” in *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, 2017
- [13] M. G. Bechtel, E. McElhiney, M. Kim, and H. Yun, “DeepPicar: A Low-cost Deep Neural Network-based Autonomous Car,” in *IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, 2018.
- [14] L. Itti, C. Koch, and E. Niebur, “A Model of Saliency-Based Visual Attention for Rapid Scene Analysis,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 11, pp. 1254–1259, 1998.
- [15] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, “Learning Deep Features for Discriminative Localization,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 2921–2929.
- [16] J. Long, E. Shelhamer, and T. Darrell. “Fully convolutional networks for semantic segmentation,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.
- [17] K. Simonyan and A. Zisserman, “VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION,” in *International Conference on Learning Representations*, 2015.
- [18] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, “Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization,” in *IEEE International Conference on Computer Vision (ICCV)*, 2017.
- [19] M. D. Zeiler and R. Fergus, “Visualizing and Understanding Convolutional Networks,” in *European Conference on Computer Vision*, 2014, pp. 818–833.
- [20] K. Simonyan, A. Vedaldi, and A. Zisserman, “Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps,” in *IEEE International Conference on Computer Vision (ICCV)*, 2014.

ACKNOWLEDGEMENT

First of all, I sincerely extend my heartfelt gratitude to my advisor, Professor. HungSun Son. From the beginning of my master's degree course, he has provided all the supports as well as allowed opportunities to study for me. With his consideration, I was able to experience on various fields and could tried on my research. Especially, for this work, Prof Son supervised academic approach and all the description following logical flow with his academic knowledge and attitude as a scholar. I've learned so many things as much as I can't mention all of them here, from his advice and approach.

Also, I would like to appreciate to members of my research group, Electromechanical Systems and Control Laboratory (ESCL), Jiyun Jeon, Myunggun Kim, Seongmin Lee, Chanbeom Bak, Minho Shin, Sangheon Lee, Wonmo Chung, Sejoon Joo and Hoyoung Kim. Despite the differences in areas of research, I felt free to get and share advice and opinions with them. They encouraged me, and I got inspiration from them when I was faced various problem. I think that I was very fortunate to be with them and I won't forget all.

Lastly, Thanks to god and my family, I might not have finished this research without their moral support at a time when the work made me hard.

I'm grateful to all those who have been involved in this process, because I couldn't have done all these works by myself. Therefore, I will do my best in the future to repay this kindness.